Robust and Efficient Machine Learning under Data Uncertainty: From Loss Functions to Data-Aware Representations

Ph.D. Thesis

By

Mushir Akhtar

under the supervision of

Prof. M. Tanveer

Dr. Mohd. Arshad



Department of Mathematics

INDIAN INSTITUTE OF TECHNOLOGY INDORE

September 2026

Robust and Efficient Machine Learning under Data Uncertainty: From Loss Functions to Data-Aware Representations

A THESIS

submitted in partial fulfillment of the
requirements for the award of the degree
of
DOCTOR OF PHILOSOPHY

By
Mushir Akhtar

under the supervision of
Prof. M. Tanveer
Dr. Mohd. Arshad



Department of Mathematics
INDIAN INSTITUTE OF TECHNOLOGY INDORE
September 2026

INDIAN INSTITUTE OF TECHNOLOGY INDORE

I hereby certify that the work which is being presented in the thesis entitled **Robust and Efficient Machine Learning under Data Uncertainty: From Loss Functions to Data-Aware Representations** in the partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in the **Department of Mathematics, Indian Institute of Technology Indore**, is an authentic record of my own work carried out under the supervision of **Prof. M. Tanveer and Dr. Mohd. Arshad, Indian Institute of Technology Indore**.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Mushir Akhtar
10-09-26

Signature of the student with date

(Mushir Akhtar)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Arshad
10/09/26

10/09/2026

Signature of thesis supervisors with date

(Prof. M. Tanveer)

(Dr. Mohd. Arshad)

The Ph.D. oral examination details will be updated after the defense.

Arshad
10/09/26

10/09/2026

Signature of thesis supervisors with date

(Prof. M. Tanveer)

(Dr. Mohd. Arshad)

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to all those who have contributed to making this research journey meaningful and productive. First and foremost, I extend my sincere thanks to my supervisors, Prof. M. Tanveer and Dr. Mohd. Arshad, for their constant guidance, encouragement, and support throughout my Ph.D. journey. Whatever I have learned and achieved during this period has been possible because of their mentorship, patience, and continuous motivation. I am also deeply grateful to my research progress committee members, Dr. V. K. Sohani and Dr. Nagendra Kumar, for their valuable time, insightful feedback, and constant support, which significantly helped me improve the quality of my work. I sincerely acknowledge CSIR for providing financial support during my Ph.D. program. I am thankful to all members of the OPTIMAL Research Group and the Statistical & Data Science Research Group for their stimulating discussions, timely help, and collaborative spirit throughout this journey. I also extend my gratitude to the Department of Mathematics staff, as well as the administrative and support staff of the institute, for their assistance and cooperation during my Ph.D. work. Lastly, I express my deepest respect and love to my family for their constant care, patience, and unconditional support. I am also thankful to everyone who, directly or indirectly, contributed to this work.

ABSTRACT

Reliable machine learning in real-world environments is often limited not by the expressive power of a model alone, but by how the learning objective responds to corrupted labels, outliers, overlapping classes, class imbalance, ill-conditioned optimization, and arbitrary random representations. This thesis, entitled *Robust and Efficient Machine Learning under Data Uncertainty: From Loss Functions to Data-Aware Representations*, develops a coherent set of robust and computationally efficient learning strategies by redesigning four fundamental components of supervised learning: the loss function, the sample-weighting mechanism, the optimization procedure, and the initialization of randomized representations. The first contribution is the RoBoSS loss, a robust, bounded, sparse, and smooth classification loss that limits the influence of extreme samples while preserving useful margin behavior and optimization stability. RoBoSS is integrated into support vector machine (SVM) learning and further extended to twin support vector machine (TSVM) learning through the RoBoSS-based Twin Support Vector Machine (RoBoTS) and its inverse-free variant, demonstrating that bounded sparse loss design can improve robustness without sacrificing efficiency. The thesis then develops the Wave loss function, a bounded, smooth, and asymmetric classification loss, and embeds it into SVM and TSVM frameworks to obtain Wave-SVM and Wave-TSVM for learning under noisy and uncertain data. For regression, the HawkEye loss is introduced to combine boundedness, smoothness, and an insensitive zone, leading to an inverse-free HawkEye least-squares support vector regression (HE-LSSVR) model capable of controlling large residuals while ignoring practically negligible deviations. Beyond loss design, the thesis addresses class-imbalanced and uncertain data through RAPID, a radius-adaptive, probability-aware, and class-imbalance-driven weighting scheme that integrates class-center proximity, local class consistency, and imbalance correction within least-squares SVM learning for robust classification, including Alzheimer's disease diagnosis. The thesis also extends robust objective design to randomized neural networks by developing XG-RVFL and Wave-BLS, which replace squared-error-dominated formulations with bounded loss-based objectives and avoid direct matrix inversion, thereby improving stability under noise and outliers while retaining the speed of

randomized learning. Finally, the thesis introduces Copula-Aligned Weight Initialization (CAWI), a dependency-aware initialization strategy that samples frozen random weights according to copulas fitted to the empirical dependence structure of input features, improving randomized representations without changing the closed-form or backpropagation-free training paradigm. Across benchmark, noisy, outlier-contaminated, class-imbalanced, and biomedical datasets, supported by appropriate statistical tests, the proposed methods consistently improve robustness, predictive performance, and computational reliability. Collectively, the thesis establishes that robust and efficient learning can be achieved not by replacing established models wholesale, but by principled redesign of the mechanisms that determine sample influence, error penalization, numerical stability, and random feature construction.

Keywords: Robust learning, loss functions, support vector machines, twin support vector machines, class imbalance, randomized neural networks, broad learning system, copula-based initialization, biomedical diagnosis.

List of Publications

Publications from the Thesis

Journal Papers

J1. **Mushir Akhtar**, M. Tanveer and M. Arshad. RoBoSS: A Robust, Bounded, Sparse, and Smooth Loss Function for Supervised Learning. ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, 47(1):149–160, 2025. doi:10.1109/TPAMI.2024.3465535. (Received Institute Best Research Paper Award, IIT Indore, 2025).

J2. **Mushir Akhtar**, M. Tanveer and M. Arshad. RoBoTS: A Robust Bounded Twin SVM Based on RoBoSS Loss Function. ***Pattern Recognition, Elsevier***, 179(B):113653, 2026. doi:10.1016/j.patcog.2026.113653.

J3. **Mushir Akhtar**, M. Tanveer, M. Arshad and Alzheimer's Disease Neuroimaging Initiative. Advancing Supervised Learning with the Wave Loss Function: A Robust and Smooth Approach. ***Pattern Recognition, Elsevier***, 155:110637, 2024. doi:10.1016/j.patcog.2024.110637.

J4. **Mushir Akhtar**, M. Tanveer and M. Arshad. HawkEye: A Robust Loss Function for Regression with Bounded, Smooth, and Insensitive Zone Characteristics. ***Applied Soft Computing, Elsevier***, 176:113118, 2025. doi:10.1016/j.asoc.2025.113118.

J5. **Mushir Akhtar**, A. Quadir, M. Tanveer and M. Arshad. Dual-Center RAPID-LSSVM: Radius-Adaptive, Probability and Imbalance Driven Weighting for Alzheimer's Diagnosis. ***Neural Networks, Elsevier***, 202:108956, 2026. doi:10.1016/j.neunet.2026.108956.

J6. **Mushir Akhtar**, A. Kumari, M. Sajid, A. Quadir, M. Arshad, P. N. Suganthan and M. Tanveer. Towards Robust and Inversion-Free Randomized Neural Networks: The XG-RVFL Framework. ***Pattern Recognition, Elsevier***, 172(D):112711, 2026. doi:10.1016/j.patcog.2025.112711.

Conference Papers

C1. **Mushir Akhtar**, M. Tanveer and M. Arshad. CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks. ***International Conference on Artificial Intelligence and Statistics (AISTATS)***, 2026, Tangier, Morocco. url:https://openreview.net/forum?id=fwqydIjLn6.

C2. **Mushir Akhtar**, A. Varshney, A. Quadir, A. Rahaman, M. Arshad and M. Tanveer. Robust Broad Learning System with Wave Loss for Classification under Data Uncertainty. ***IEEE World Congress on Computational Intelligence (WCCI)***, 2026. (Accepted).

Publications Other Than the Thesis

Journal Papers

OJ1. K. Ali, **Mushir Akhtar**, A. Zafar and M. Tanveer. Intuitionistic Fuzzy and Robust Loss Fused Framework for Stable and Efficient RVFL Learning. ***IEEE Transactions on Fuzzy Systems***. 2026. (Accepted).

OJ2. A. Kumari, **Mushir Akhtar**, M. Tanveer and M. Arshad. Diagnosis of Breast Cancer Using Flexible Pinball Loss Support Vector Machine. ***Applied Soft Computing, Elsevier***, 157:111454, 2024. doi:10.1016/j.asoc.2024.111454.

OJ3. A. Kumari, **Mushir Akhtar**, R. Shah and M. Tanveer. Support Matrix Machine: A Review. ***Neural Networks, Elsevier***, 181:106767, 2025. doi:10.1016/j.neunet.2024.106767.

OJ4. A. Quadir, **Mushir Akhtar** and M. Tanveer. Enhancing Multiview Synergy: Robust Learning by Exploiting the Wave Loss Function with Consensus and Complementarity Principles. ***Neural Networks, Elsevier***, 188:107433, 2025. doi:10.1016/j.neunet.2025.107433.

OJ5. M. Tanveer, M. Sajid, **Mushir Akhtar**, et al. Fuzzy Deep Learning for the Diagnosis of Alzheimer's Disease: Approaches and Challenges. ***IEEE Transactions on Fuzzy Systems***, 32(10):5477–5492, 2024. doi:10.1109/TFUZZ.2024.3409412.

OJ6. M. Tanveer, A. Tiwari, **Mushir Akhtar** and C. T. Lin. Enhancing Imbalance Learning: A Novel Slack-Factor Fuzzy SVM Approach. ***IEEE Transactions on Emerging Topics in Computational Intelligence***, 9(4):3112–3121, 2025. doi:10.1109/TETCI.2024.3524718.

Conference Papers

OC1. **Mushir Akhtar**, M. Tanveer and M. Arshad. Residual-Guided Randomized Neural Networks. ***IEEE World Congress on Computational Intelligence (WCCI)***, 2026.

(Accepted).

OC2. **Mushir Akhtar**, M. Tanveer and M. Arshad. GL-TSVM: A Robust and Smooth Twin Support Vector Machine with Guardian Loss Function. ***Pattern Recognition, ICPR 2024, Lecture Notes in Computer Science***, 15302:63–78, 2025. doi:10.1007/978-3-031-78166-7_5.

OC3. **Mushir Akhtar**, R. Mishra, M. Tanveer and M. Arshad. Advancing RVFL Networks: Robust Classification with the HawkEye Loss Function. ***Neural Information Processing, ICONIP 2024, Lecture Notes in Computer Science***, 15288:226–240, 2025. doi:10.1007/978-981-96-6582-2_16.

OC4. A. Varshney*, **Mushir Akhtar***, M. Arshad and M. Tanveer. Metric-Enhanced Hybrid Kernel Probabilistic Neural Networks for Robust Classification. ***IEEE World Congress on Computational Intelligence (WCCI)***, 2026. (Accepted, *equal contribution).

OC5. M. Sajid*, **Mushir Akhtar***, M. Tanveer and S. Mitra. Fuzzy Learning at 60: Future of Trustworthy AI in Healthcare and LLM. ***IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)***, 2025. (*equal contribution).

OC6. R. Mishra, **Mushir Akhtar** and M. Tanveer. CI-RKM: A Class-Informed Approach to Robust Restricted Kernel Machines. ***International Joint Conference on Neural Networks (IJCNN)***, 2025.

OC7. M. Sajid, **Mushir Akhtar**, A. Quadir and M. Tanveer. RVFL-X: A Novel Randomized Network Based on Complex Transformed Real-Valued Tabular Datasets. ***International Joint Conference on Neural Networks (IJCNN)***, 2025.

OC8. A. Kumari, **Mushir Akhtar**, M. Tanveer and P. N. Suganthan. R^2VFL: A Robust Random Vector Functional Link Network with Huber-Weighted Framework. ***International Joint Conference on Neural Networks (IJCNN)***, 2025.

OC9. A. Quadir, A. Rahaman, **Mushir Akhtar** and M. Tanveer. Robust Dual-Model Collaborative Random Vector Functional Link Network. ***IEEE World Congress on***

Computational Intelligence (WCCI), 2026. (Accepted).

OC10. A. Quadir, M. Sajid, **Mushir Akhtar** and M. Tanveer. Twin Restricted Kernel Machines for Multiview Classification. ***International Joint Conference on Neural Networks (IJCNN)***, 2025.

OC11. A. Rahaman, A. Quadir, M. Sajid, **Mushir Akhtar** and M. Tanveer. ECA-BLS: An Efficient Complex-Augmented Broad Learning System. ***IEEE World Congress on Computational Intelligence (WCCI)***, 2026. (Accepted).

Contents

List of Figures xi

List of Tables xxiii

List of Abbreviations xxiv

1 Introduction 1

1.1 Learning as an Interaction Between Loss, Weights, and Representation . . . 4

1.2 Role of Loss Functions in Learning 5

1.2.1 Loss Functions as Influence Mechanisms 6

1.2.2 Classification Losses and Decision Boundaries 7

1.2.3 Regression Losses and Residual Control 8

1.2.4 Desirable Properties of Loss Functions 9

1.3 Why Hyperplane-Based Models? 10

1.3.1 Margin Principle and Generalization 10

1.3.2 Least-Squares SVM and the Efficiency-Robustness Trade-Off . . . 11

1.3.3 Twin SVM and Class-Specific Geometry 11

1.3.4 SVR and Robust Regression 12

1.3.5 Why Hyperplane Models Remain Central 12

1.4 Effect of Weighting Schemes in Learning 13

1.4.1 Weighted Empirical Risk 13

1.4.2 Class Imbalance 14

1.4.3 Uncertainty and Local Reliability 14

1.5 Randomized Neural Networks as Efficient Alternatives to Fully Trained Neural Networks 15

1.5.1 Basic Idea of Randomized Neural Networks 15
1.5.2 Random Vector Functional Link, Extreme Learning Machine, Broad Learning System, and Deep RVFL 16
1.5.3 Benefits Over Fully Trained Neural Networks 17
1.5.4 Limitations of Randomized Neural Networks 18
1.6 Noise, Imbalance, and Randomness as a Unified Problem 19
1.7 Research Gaps 20
1.7.1 Gap 1: Robust Classification Losses Need a Better Balance of Boundedness, Smoothness, Sparsity, and Asymmetry 20
1.7.2 Gap 2: Robust Regression Requires Boundedness Together with an Insensitive Zone 21
1.7.3 Gap 3: Class Imbalance and Sample Uncertainty Are Often Treated Separately 21
1.7.4 Gap 4: Randomized Neural Networks Need Robust and Inversion-Free Learning 21
1.7.5 Gap 5: Independent Random Initialization Ignores Feature Dependence 22
1.8 Scope of the Thesis 22
1.9 Research Objectives 23
1.10 Major Contributions 24
1.11 Thesis Organization 26

2 Literature Review **29**
2.1 Notation and Problem Setting 29
2.2 Margin-Based Learning 31
2.2.1 Support Vector Machines 31
2.2.2 Least-Squares SVM 33
2.2.3 Twin Support Vector Machines 33
2.2.4 Support Vector Regression 34
2.3 Classification Loss Functions 35

2.3.1 Convex but Non-Smooth Losses: Hinge and Pinball 36
2.3.2 Convex and Smooth Losses: Squared Hinge, Smooth Pinball, and LINEX 39
2.3.3 Non-Convex and Non-Smooth Losses: Ramp and Truncated Pinball 40
2.3.4 Deep-Learning Losses for Class Imbalance and Label Noise 42
2.3.5 Automated Loss-Function Discovery 44
2.4 Regression Loss Functions 45
2.4.1 Unbounded Regression Losses 45
2.4.2 Bounded and Insensitive Regression Losses 48
2.4.3 Smooth Bounded Regression Losses 51
2.4.4 Broader Robust-Regression Perspective 52
2.5 Class Imbalance, Noise, and Sample Weighting 54
2.5.1 Data-Level and Algorithm-Level Strategies 54
2.5.2 Classical Fuzzy Membership Schemes 55
2.5.3 Limitations of Distance-Based Weighting 57
2.6 Randomized Neural Networks 58
2.6.1 Random Vector Functional Link Networks and Extreme Learning Machines 58
2.6.2 Broad Learning System 59
2.6.3 Deep RVFL 61
2.7 Copula-Based Dependency Modeling 63
2.8 Optimization and Numerical Stability 65
2.9 Evaluation Metrics 66
2.9.1 Classification Metrics 66
2.9.2 Regression Metrics 67
2.10 Statistical Comparison of Learning Algorithms 67
2.10.1 Friedman and Iman–Davenport Tests 68
2.10.2 Nemenyi Post-Hoc Test 69
2.10.3 Wilcoxon Signed-Rank Test 69
2.10.4 Win–Tie–Loss Test 70

2.11 Positioning of the Thesis 71

3 RoBoSS: A Robust, Bounded, Sparse, and Smooth Loss Function for Supervised Learning 75

3.1 Introduction and Motivation 76
3.2 RoBoSS Loss and RoBoSS-SVM Formulation 78
3.3 Theoretical Analysis of the RoBoSS Loss 83
3.4 Optimization of RoBoSS-SVM 87
3.4.1 Computational Analysis 91
3.5 RoBoSS-SVM Experimental Evaluation 91
3.5.1 Experimental Setup and Parameter Selection 91
3.5.2 Evaluation on KEEL and UCI Datasets 94
3.5.3 Evaluation on Datasets with Introduced Outliers and Label Noise . . 99
3.5.4 Evaluation on Biomedical Datasets 102
3.5.5 Statistical Analysis of RoBoSS-SVM Results 104
3.5.6 Sensitivity Analysis of RoBoSS-SVM 107
3.6 From RoBoSS-SVM to RoBoTS 109
3.7 Twin SVM Background and Related Losses 109
3.7.1 Notation and Modeling Link 109
3.8 RoBoTS and Inverse-Free RoBoTS Models 110
3.8.1 Formulation and Optimization of RoBoTS Model 110
3.8.1.1 **Linear Formulation** 110
3.8.1.2 **Nonlinear Formulation** 113
3.8.2 Inverse-Free RoBoTS (i-RoBoTS): An Alternate Optimization Approach 118
3.8.2.1 **Linear Formulation** 118
3.8.2.2 **Nonlinear Formulation** 120
3.8.3 Generalization Analysis of RoBoTS Model 121
3.8.4 Computational Complexity 123
3.9 RoBoTS Experimental Evaluation 124

3.9.1 Compared Models, Experimental Setup, and Hyperparameter Tuning 124
3.9.2 Experimental Results on Benchmark KEEL and UCI Datasets . . . 129
3.9.3 Experimental Results on Biomedical Datasets 135
3.9.4 Experimental Results on Noisy Datasets 139
3.9.5 Sensitivity Analysis of the RoBoSS Loss Hyperparameters 141
3.9.6 Effectiveness of the Proposed RoBoTS Model 143
3.10 Chapter Summary 143

4 Advancing Supervised Learning with the Wave Loss Function: A Robust and Smooth Approach 145

4.1 Introduction and Motivation 146
4.2 Wave Loss and Proposed Models 148
4.2.1 Wave Loss Function 148
4.2.2 Smoothness of the Wave Loss Function 150
4.2.3 Theoretical Analysis of the Wave Loss Function 151
4.2.4 Formulation of Wave-SVM 155
4.2.4.1 Adam for Linear Wave-SVM 157
4.2.4.2 Adam for Nonlinear Wave-SVM 158
4.2.5 Formulation of Wave-TSVM 160
4.2.5.1 Linear Wave-TSVM 161
4.2.5.2 Nonlinear Wave-TSVM 163
4.2.6 Computational Complexity 166
4.3 Numerical Experiments 167
4.3.1 Experimental Setup and Parameter Selection 167
4.3.2 Evaluation on UCI and KEEL Datasets 171
4.3.3 Evaluation on UCI and KEEL Datasets with Gaussian Noise 177
4.3.4 Results Analysis on ADNI Dataset 180
4.4 Statistical Analysis of Wave-SVM and Wave-TSVM 181
4.5 Chapter Summary 186

5 HawkEye: A Robust Loss Function for Regression with Bounded, Smooth, and Insensitive Zone Characteristics 189

5.1 Introduction . . . 190

5.2 Related Work . . . 193

5.3 HawkEye Loss and HE-LSSVR . . . 193

5.3.1 HawkEye Loss Function . . . 193

5.3.2 Proofs of HawkEye Loss Properties . . . 196

5.3.3 Proof of the Properties of the HawkEye Loss Function . . . 196

5.3.4 Comparative Analysis of HawkEye Loss Function with Baseline Loss Functions . . . 201

5.3.5 SVR with HawkEye Loss Function . . . 205

5.3.6 Adaptive Moment Estimation (Adam) Algorithm for HE-LSSVR . . 206

5.4 Numerical Experiments and Result Analysis . . . 212

5.4.1 Evaluation Metrics . . . 212

5.4.2 Experiments on Benchmark UCI Datasets . . . 213

5.4.3 Experiments on Synthetic Datasets . . . 220

5.4.4 Experiments on Time Series Datasets . . . 224

5.4.5 Experiments on Brain Age Dataset . . . 224

5.5 Chapter Summary . . . 226

6 Dual-Center RAPID-LSSVM: Radius-Adaptive, Probability, and Class-Imbalance-Driven Weighting for Alzheimer's Diagnosis 229

6.1 Introduction . . . 230

6.2 Related Work . . . 233

6.3 RAPID Weighting Scheme and RAPID-LSSVM . . . 233

6.3.1 Limitations of Existing Fuzzy Schemes . . . 234

6.3.2 Proposed RAPID Weighting Scheme . . . 235

6.3.2.1 Radius-Adaptive Proximity Mechanism . . . 235

6.3.2.2 Probability-Aware Refinement . . . 236

6.3.2.3 Imbalance-Ratio Correction . . . 237
6.3.3 Computational Complexity . . . 239
6.3.4 Proposed RAPID-LSSVM . . . 241
6.4 Experimental Results . . . 243
6.4.1 Experimental Setup . . . 243
6.4.2 Evaluation on KEEL and UCI Datasets . . . 245
6.4.3 Evaluation on KEEL and UCI Datasets with Added Label Noise . . . 248
6.4.4 Ablation Study . . . 250
6.4.5 Evaluation on ADNI Dataset . . . 252
6.4.6 Sensitivity Analysis . . . 254
6.4.7 Practical Scope and Limitations . . . 259
6.5 Chapter Summary . . . 260

7 Robust and Inversion-Free Learning in Randomized Neural Networks 261

7.1 XG-RVFL Framework . . . 262
7.1.1 Introduction . . . 262
7.1.2 Background and Notation . . . 265
7.1.3 Proposed XG-RVFL Framework . . . 266
7.1.3.1 FleXi Guardian (XG) Loss . . . 267
7.1.3.2 Theoretical Properties of the FleXi Guardian (XG) Loss . . . 269
7.1.3.3 Formulation and Optimization of XG-RVFL . . . 271
7.1.3.4 Generalization Bound for the Proposed XG-RVFL . . . 277
7.1.3.5 Computational Complexity . . . 279
7.1.4 Experiments, Results and Discussion . . . 280
7.1.4.1 Evaluation on UCI Binary Datasets . . . 282
7.1.4.2 Evaluation on UCI Multiclass Datasets . . . 286
7.1.4.3 Evaluation on KEEL Datasets . . . 289
7.1.4.4 Sensitivity Analysis . . . 291
7.2 Wave-BLS Framework . . . 293

7.2.1 Background and Notation . . . 293
7.2.2 Proposed Wave-BLS Framework . . . 294
7.2.2.1 Wave Loss Function . . . 294
7.2.2.2 Formulation & Optimization: Broad Learning System with Wave Loss Function (Wave-BLS) . . . 295
7.2.3 Wave-BLS Experiments and Discussion . . . 298
7.2.3.1 Experimental Setup . . . 299
7.2.3.2 Robustness Analysis under Noise and Outliers . . . 302
7.3 Statistical Significance Analysis . . . 303
7.4 Sensitivity Analysis of Wave Loss Hyperparameters . . . 304
7.5 Computational Trade-Offs and Evaluation Scope . . . 306
7.6 Chapter Summary . . . 307

8 CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks 309

8.1 Introduction . . . 310
8.2 Preliminaries . . . 313
8.2.1 Randomized Neural Network (RdNN) Setting . . . 313
8.2.2 Copulas . . . 314
8.2.2.1 Copula . . . 314
8.2.2.2 Invariance . . . 314
8.2.2.3 Sklar's Theorem . . . 315
8.3 Problem Statement . . . 315
8.3.1 Limitation . . . 316
8.4 Method . . . 317
8.4.1 Step 1: Probability-Integral Transform of Features . . . 317
8.4.2 Step 2: Fit a Multivariate Copula on U . . . 318
8.4.3 Step 3: Sample Dependence-Aligned Weight Columns . . . 318
8.4.4 Step 4: Closed-Form Training (Unchanged) . . . 319
8.4.5 Copula Families: Parameterization and Estimation . . . 319

8.4.5.1 Elliptical Families (Gaussian, t) . . . 320
8.4.5.2 Archimedean Families (Clayton, Gumbel, Frank) . . . 320
8.4.6 Computational Complexity of CAWI . . . 321
8.4.6.1 Step 1: Probability-Integral Transform . . . 321
8.4.6.2 Step 2: Copula Fitting . . . 322
8.4.6.3 Step 3: Sampling Dependence-Aligned Weight Columns . 322
8.4.6.4 Step 4: Closed-Form RdNN Training (Unchanged) . . . 322
8.4.6.5 Summary . . . 323
8.5 Empirical Results . . . 323
8.5.1 Dataset Description . . . 325
8.5.2 Performance Evaluation . . . 326
8.5.2.1 Results on BreaKHis Dataset . . . 326
8.5.2.2 Results on Schizophrenia Dataset . . . 327
8.5.3 Results on 83 UCI Datasets . . . 328
8.6 Experimental Setup and Hyperparameter Setting . . . 328
8.7 Multi-Seed Stability Analysis . . . 333
8.8 Statistical Significance Across Datasets . . . 335
8.9 Empirical Training-Time Analysis . . . 336
8.10 Chapter Summary . . . 336

9 Conclusions and Future Directions 339

9.1 Synthesis of Contributions . . . 339
9.2 Major Findings . . . 343
9.3 Future Directions . . . 344
9.3.1 Extending Robust Loss Functions to Other Learning Paradigms . . . 345
9.3.2 Direct Multiclass Extensions of SVM and TSVM Models . . . 345
9.3.3 Automatic Hyperparameter and Loss-Shape Adaptation . . . 346
9.3.4 Robust Deep and Ensemble Randomized Neural Networks . . . 346
9.3.5 Automatic Copula Selection for CAWI . . . 347
9.3.6 Theory for Robust and Randomized Learning . . . 348

9.4 Concluding Perspective 348

List of Figures

2.1 Geometric view of SVM. The central hyperplane separates the two classes, while the two dashed hyperplanes define the unit-margin band. Samples on or inside this band determine the solution and become support vectors. . . . 32
2.2 Geometric view of TSVM. Instead of a single separating hyperplane, TSVM learns two nonparallel class-specific hyperplanes and classifies a test sample using its relative distance to these hyperplanes. 34
2.3 Geometric interpretation of SVR. Samples inside the ε-tube incur no loss, while samples outside the tube are penalized through slack variables. 35
2.4 Hinge loss. The loss is zero for correctly classified samples outside the margin and grows linearly for margin violations. 38
2.5 Pinball loss. The parameter τ controls the penalty assigned to samples on the correctly classified side of the margin. 38
2.6 Squared hinge loss. The quadratic penalty emphasizes large margin violations, which can improve fitting but increases outlier sensitivity. 39
2.7 Smooth pinball loss. The parameter τ controls asymmetric quadratic penalties on the two sides of the margin. 40
2.8 LINEX loss. The parameter a controls asymmetric growth, allowing one side of the margin violation to be penalized more heavily. 41
2.9 Ramp loss. Hard truncation caps the influence of severe margin violations but introduces non-smooth points. 41
2.10 Truncated pinball loss with $\tau = 0.5$, $\delta_1 = 1$, and $\delta_2 = 0.25$. The two caps bound the loss on both sides of the margin violation axis. 42
2.11 Least-squares loss. Quadratic growth makes optimization convenient but gives large residuals excessive influence. 46

2.12 Absolute loss. Linear growth improves robustness relative to least squares, but the cusp at zero is non-smooth. 47

2.13 Huber loss. The loss is quadratic near zero and linear for large residuals, balancing smoothness and robustness. 47

2.14 ε-insensitive loss. The zero-loss tube improves sparsity and tolerance to small deviations. 48

2.15 Ramp-insensitive least-square-type loss. The insensitive zone and upper cap improve robustness, while hard transitions introduce non-smoothness. . . . 49

2.16 Nonconvex least-squares loss. Ordinary residuals are penalized quadratically, while extreme residuals are capped. 50

2.17 Quadratic nonconvex insensitive loss. The loss combines a zero tube, quadratic middle behavior, and controlled outer growth. 50

2.18 Canal loss. The loss keeps the SVR-style zero tube and caps large residuals at a fixed level. 51

2.19 Bounded least-squares loss. Smooth saturation limits the effect of large residuals without using hard truncation. 52

2.20 BLINEX loss for different values of a and θ. The parameter a controls asymmetry, while θ changes the saturation behavior. 52

2.21 Challenges for fuzzy weighting in real-world data. (a) Many valid samples can be far from the class center. (b) Central samples may deserve similar reliability despite small distance differences. (c) Noisy samples can share the same center distance as clean samples, requiring information beyond distance alone. 57

2.22 RVFL architecture. Hidden-layer parameters are randomly sampled and fixed, while only the output weights are trained. Direct input-to-output links distinguish RVFL from ELM. 60

2.23 BLS architecture. Feature mapping nodes and enhancement nodes are generated in a broad flat structure, then concatenated to train the output weights. 62

2.24 Deep RVFL architecture. Multiple random hidden layers are stacked, and the output layer is trained using an augmented representation containing the input and hidden-layer outputs. 63

2.25 Copula decomposition. The empirical variables are transformed through their marginal CDFs to uniform variables, and the copula models the remaining dependence structure on the unit square. 64

3.1 RoBoSS loss behavior: (a) fixed $\lambda = 1.5$ with different values of a, and (b) fixed $a = 5$ with different values of λ. 77

3.2 Demonstrate the graph of $\int_Y \mathcal{L}_{RoBoSS}\left(1 - yf(x)\right) d\mathcal{P}(y|x)$ with respect to $f(x)$ for different $P(x)$ values. (a) For $P(x) > 1/2$ and (b) for $P(x) < 1/2$. 85

3.3 Flowchart of the proposed $\mathcal{L}_{RoBoSS}$-SVM model. It depicts the essential stages of the proposed $\mathcal{L}_{RoBoSS}$-SVM, demonstrating the process from raw input data to the final decision function. This flowchart shows the incorporation of the RoBoSS loss function and illustrates the use of the representer theorem and the NAG optimization algorithm. 89

3.4 Sensitivity analysis of the $\mathcal{L}_{RoBoSS}$-SVM model with respect to the loss hyperparameters a and λ on abalone9-18, echocardiogram, titanic, and ecoli3 datasets. The 3D surface plots illustrate the model's accuracy variations as a and λ are systematically adjusted while keeping other hyperparameters fixed at their optimal values. 108

3.5 Sensitivity analysis of the RoBoTS model with respect to the loss hyperparameters a and η, presented through 3D surface plots for four datasets: (a) horse_colic, (b) monk3, (c) congressional_voting, and (d) yeast-0-3-5-9_vs_7-8. The color scale represents accuracy values, with warmer colors (yellow–red) indicating higher accuracy and cooler colors (blue–green) indicating lower accuracy. 141

4.1 Illustration of wave loss function for fixed $\lambda = 1$ and different values of a. Subfigures (a), (b), (c), and (d) demonstrate that the value of a controls the strength of the penalty for correctly classified and misclassified samples. . . 152

4.2 Illustrate the plot of $\int_Y \mathfrak{L}_{wave}\,(1 - yf(x))\,d\mathsf{P}(y|x)$ over $f(x)$ with varying values of $\mathsf{P}(x)$. (a) Depict the case where $\mathsf{P}(x)$ is greater than $1/2$, and (b) illustrate the scenario where $\mathsf{P}(x)$ less than $1/2$. 155

5.1 Illustration of the HawkEye loss function and its gradient for varying values of a and λ. The loss function plots demonstrate that for a fixed $\lambda = 1$ and $\varepsilon = 0.5$, increasing a results in a steeper transition beyond the insensitive zone, making the function more sensitive to larger residuals. Conversely, for a fixed $a = 5$ and $\varepsilon = 0.5$, increasing λ raises the upper bound of the loss, effectively constraining the maximum penalty for large errors. The gradient plots further illustrate these effects, where a higher a leads to sharper changes in gradient magnitude, reinforcing stronger penalization for deviations, while increasing λ scales the gradient without altering its fundamental behavior. These visualizations highlight how adjusting a and λ enables fine-tuning of the loss function, balancing robustness and sensitivity to optimize performance in different tasks. 195

5.2 Visual comparison of HawkEye loss and baseline loss functions. Subfigures (a), (b), (c), and (d) demonstrate the comparison of HawkEye loss function with the least square loss, insensitive loss, canal loss, and bounded least square loss, respectively. 202

5.3 Flowchart illustrating the key stages of the HE-LSSVR model, showcasing the transformation from original input data to the decision function, highlighting the strategic integration of the HawkEye loss function and the application of representer theorem and Adam algorithm. 211

6.1 (a) Depicts a sample distribution where the majority of samples are dispersed farther from the class-center. (b) Illustrates a sample distribution with the majority of samples densely clustered around the class-center. (c) Demonstrates that noise samples are positioned at similar distances from the class-center as the normal samples, highlighting the challenge of distinguishing noise from legitimate data points based solely on their proximity to the center. 234

6.2 Illustration of the proposed RAPID mechanism. (a) Radius-adaptive proximity weighting showing the effect of the flexibility parameter λ. (b) End-to-end RAPID-LSSVM pipeline integrating proximity, probability, and imbalance-driven weighting into the learning framework. 238

6.3 Illustrates the impact of various levels of label noise on the performance of the proposed RAPID-LSSVM models compared to the baseline models, demonstrating their robustness against label noise. The datasets used are (a) ecoli-0-1-4-6_vs_5, (b) ecoli-0-3-4-7_vs_5-6, (c) votes, and (d) yeast3. . . . 255

6.4 Demonstrates the effect of the regularization parameter C and the kernel parameter σ on the performance of the proposed RAPID-LSSVM models. The datasets used for this analysis are (a) brwisconsin with RAPID-LSSVM-I, (b) breast_cancer_wisc with RAPID-LSSVM-I, (c) brwisconsin with RAPID-LSSVM-II, and (d) breast_cancer_wisc with RAPID-LSSVM-II. The surface plots illustrate how changes in C and σ influence the accuracy of the models, highlighting the sensitivity of the RAPID-LSSVM models to these parameters. 256

6.5 Explores the impact of the flexible parameter λ and the k-nearest neighborhood parameter k on the performance of the proposed RAPID-LSSVM models. The datasets analyzed are (a) ecoli-0-6-7_vs_3-5 with RAPID-LSSVM-I, (b) statlog-german-credit with RAPID-LSSVM-I, (c) ecoli-0-6-7_vs_3-5 with RAPID-LSSVM-II, and (d) statlog-german-credit with RAPID-LSSVM-II. The surface plots illustrate how variations in λ and k parameters affect the models' accuracy, highlighting the sensitivity and influence of these parameters on the performance of the RAPID-LSSVM models. 257

7.1 Illustrate the behavior of the proposed XG loss function under varying parameter configurations. Subfigures (a) and (b) are plotted for fixed values of a ($a = 1$ in (a) and $a = -1$ in (b)) while varying the bound-control parameter η. Subfigures (c) and (d) are plotted for a fixed $\eta = 1/2$ while varying a. Subfigure (c) explores positive values of a, showcasing its effect on the steepness and penalties for positive deviations, whereas subfigure (d) examines negative values of a, highlighting its influence on negative deviations. . 268

7.2 Sensitivity analysis of the XG-RVFL model with respect to the loss hyperparameters a and η, presented through 3D surface plots for six datasets: (a) blood, (b) breast_cancer, (c) hepatitis, (d) iris, (e) wine_quality_red, and (f) vertebral_column_3clases. The plots illustrate the effect of tuning a and η on model accuracy while keeping other hyperparameters fixed at their optimal values, highlighting the significance of the loss hyperparameters in achieving robust performance. 291

7.3 Illustration of wave loss function, showing effect of asymmetry parameter a (positive, negative, and $a = 0$) and the bounding parameter λ. 295

7.4 Sensitivity analysis of the wave loss hyperparameters in the proposed Wave-BLS model on the breast_cancer dataset. (a) Effect of the bounding parameter λ, (b) effect of the shape parameter a, and (c) their joint influence on classification accuracy. 304

8.1 Schematic of a copula. *Left:* empirical sample (with its joint CDF inset). *Middle:* univariate CDFs F_1 and F_2 (mapping $x_j \mapsto u_j = F_j(x_j)$). *Right:* the copula surface $C(u_1, u_2)$, i.e., the joint CDF on $[0, 1]^2$ with uniform marginals. The equality sign indicates Sklar's factorization $F(x_1, x_2) = C(F_1(x_1), F_2(x_2))$; the $\oplus$ highlights that C combines the marginal information into a dependence structure. 315

List of Tables

2.1 Core notation used throughout the thesis. 31

2.2 Summary of representative classification loss functions for margin-based learning. 37

2.3 Representative probability-based losses for deep classification under class imbalance or label noise. 43

2.4 Summary of representative regression loss functions. 46

2.5 Representative robust regression losses beyond the immediate SVR literature. 53

2.6 Comparative positioning of the thesis within the relevant learning literature. 73

3.1 Illustrates the key attributes of different state-of-the-art loss functions with the proposed RoBoSS loss function, highlighting their robustness, sparsity, boundedness, convexity, and smoothness. 81

3.2 The average classification accuracies, training times, and ranks of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on each 79 D1 category UCI and KEEL datasets. 95

3.2 The average classification accuracies, training times, and ranks of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on each 79 D1 category UCI and KEEL datasets. 96

3.3 The classification accuracies and training times of the $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on 9 D2 category KEEL and UCI datasets. . . . 98

3.4 The classification accuracy of the proposed $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on datasets with varying levels of outliers. 101

3.5 The classification accuracy of the proposed $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on datasets with varying levels of label noise. 101

3.6 The classification accuracies and training times of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on the EEG dataset. 103

3.7 The classification accuracies and training times of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on the BreaKHis dataset. 104

3.8 Illustrate the results of the Friedman test on D1 category UCI and KEEL datasets, the EEG dataset, and the BreaKHis dataset. 106

3.9 Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on D1 category UCI and KEEL datasets. 106

3.10 Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on the EEG dataset. 106

3.11 Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on the BreaKHis dataset. 106

3.12 Classification accuracy comparison of proposed linear RoBoTS with baseline models on UCI and KEEL datasets. 127

3.13 Classification accuracy comparison of proposed nonlinear RoBoTS with baseline models on UCI and KEEL datasets. 128

3.14 Friedman test results for linear and nonlinear RoBoTS models against baseline models on KEEL and UCI datasets. 131

3.15 Nemenyi post-hoc test results for the linear RoBoTS model against baseline models on KEEL and UCI datasets. 131

3.16 Nemenyi post-hoc test results for the nonlinear RoBoTS model against baseline models on KEEL and UCI datasets. 132

3.17 Classification accuracy comparison of proposed RoBoTS with baseline models on the BreaKHis dataset. 134

3.18 Classification accuracy comparison of proposed RoBoTS with baseline models on the ADNI dataset. 135

3.19 Classification accuracy comparison of models on noise-injected datasets. The noise level indicates the percentage of label corruption. The proposed RoBoTS model demonstrates superior robustness to increasing noise levels. 138

4.1 Comparative Analysis of SVM Models Utilizing Different Loss Functions. . 156
4.2 Description of UCI and KEEL datasets employed in the experiments. 170
4.3 Accuracy and training time for linear Wave-SVM against baseline models on UCI and KEEL datasets. 172
4.4 Accuracy and training time for non-linear Wave-SVM against baseline models using Gaussian kernel function on UCI and KEEL datasets. 174
4.5 Accuracy for non-linear Wave-TSVM against baseline models on UCI and KEEL datasets. 176
4.6 Performance comparison of the proposed Wave-SVM compare baseline models on benchmark UCI and KEEL datasets with Gaussian noise. 177
4.7 Performance comparison of the proposed Wave-TSVM compare baseline models on benchmark UCI and KEEL datasets with Gaussian noise. 179
4.8 Results of the proposed non-linear Wave-SVM against baseline models on ADNI dataset. 181
4.9 Differences in the rankings of the non-linear proposed Wave-SVM against baseline models on UCI and KEEL datasets. 185
4.10 Differences in the rankings of the non-linear proposed Wave-TSVM against baseline models on UCI and KEEL datasets. 185
4.11 Non-linear Wave-SVM results of pairwise Win-Tie-Loss test on UCI and KEEL datasets. Here, in [x/y/z], x, y, and z correspond to the count of wins, ties, and losses of the row model over column method, respectively. 185
4.12 Non-linear Wave-TSVM results of pairwise Win-Tie-Loss test on UCI and KEEL datasets. Here, in [x/y/z], x, y, and z correspond to the count of wins, ties, and losses of the row model over column method, respectively. 185

5.1 Demonstrate the characteristics of baseline loss functions used in regression and the proposed HawkEye loss. 201
5.2 The evaluation metrics. 213
5.3 Description of benchmark UCI datasets. 214

5.4 Rank of RMSE for the proposed HE-LSSVR and baseline models on benchmark UCI datasets. 214

5.5 RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the benchmark UCI datasets for the proposed HE-LSSVR model and the baseline models. 216

5.5 (Continued) RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the benchmark UCI datasets for the proposed HE-LSSVR model and the baseline models. . 217

5.6 Training time in seconds of the proposed HE-LSSVR model and baseline models on benchmark UCI datasets. 219

5.7 Illustrate the results of the Friedman test on 18 benchmark UCI datasets. . . 219

5.8 Difference in the RMSE rankings of the proposed HE-LSSVR model against baseline models on UCI datasets. 220

5.9 Description of synthetic datasets. 221

5.10 RMSE values on the synthetic datasets for the proposed HE-LSSVR model against baseline models. 222

5.11 RMSE values on the outlier added synthetic datasets for the proposed HE-LSSVR model against baseline models. 222

5.12 RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the time series datasets for the proposed HE-LSSVR model against baseline models. 223

5.13 RMSE and MAE values of the proposed HE-LSSVR model against baseline models on testing data for CH, MCI, and AD subjects. 226

6.1 Performance comparison of the proposed RAPID-LSSVM models against the baseline models on KEEL and UCI datasets without label noise. 246

6.2 Performance comparison of the proposed RAPID-LSSVM models against the baseline models on KEEL and UCI datasets with label noise. 248

6.3 Ablation study analyzing the contribution of individual components of the RAPID weighting scheme. Each variant removes one component at a time, while RAPID-LSSVM-I and RAPID-LSSVM-II correspond to the complete models with mean- and median-based centers, respectively. 251

6.4 Performance comparison between the proposed RAPID-LSSVM models and baseline models on the ADNI dataset. 254

7.1 Details of binary and multiclass datasets, including the number of samples, features, and target classes. 282
7.2 Accuracy of the proposed XG-RVFL model compared to baseline models on each of the binary datasets in categories B1 and B2. 283
7.3 Friedman test results for UCI and KEEL datasets. 286
7.4 Accuracy of the proposed XG-RVFL model compared to baseline models on each of the multiclass datasets in categories M1 and M2. 287
7.5 F1-Score of the proposed XG-RVFL model compared to baseline models on each of the KEEL datasets. 289
7.6 Dataset wise results of the proposed wave-BLS model against the baseline models on 30 UCI datasets. 300
7.7 Robustness comparison of BLS and Wave-BLS under outlier and noise contamination. 301
7.8 Statistical significance analysis using Friedman test and Nemenyi post-hoc test. . . 304

8.1 Results on the BreaKHis dataset for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 324
8.2 Results on the BreaKHis dataset for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 324
8.3 Results on the Schizophrenia ROI datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 324
8.4 Results on the Schizophrenia ROI datasets for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 325
8.5 Aggregate performance summary across 83 UCI datasets. 328

8.6 Description of binary and multiclass datasets used in the experiments. The table lists the dataset name, number of samples, number of features, and number of classes. 330
8.7 Results on the BreaKHis dataset for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 330
8.8 Results on the Schizophrenia ROI datasets for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 331
8.9 Results on benchmark binary datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 331
8.10 Results on benchmark multiclass datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline. 332
8.11 BreaKHis dataset (Seed 7): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline. 333
8.12 BreaKHis dataset (Seed 123): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline. 334
8.13 Schizophrenia ROI dataset (Seed 7): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline. . . . 334
8.14 Schizophrenia ROI dataset (Seed 123): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline. . . . 334
8.15 Average training time (in seconds) across 83 UCI datasets. 336

List of Abbreviations

AD	Alzheimer's disease	**KKT**	Karush–Kuhn–Tucker
ADNI	Alzheimer's Disease Neuroimaging Initiative	**k-NN**	k-nearest neighbour
AI	Artificial intelligence	**LSSVM**	Least-squares support vector machine
ANN	Artificial neural network	**LSSVR**	Least-squares support vector regression
BLS	Broad learning system	**LSTSVM**	Least-squares twin support vector machine
CIL	Class-imbalance learning	**ML**	Machine learning
EEG	Electroencephalography	**MSE**	Mean squared error
ELM	Extreme learning machine	**QPP**	Quadratic programming problem
ERM	Empirical risk minimization	**RdNN**	Randomized neural network
FSVM	Fuzzy support vector machine	**RBF**	Radial basis function
RVFL	Random vector functional link	**SVM**	Support vector machine
SRM	Structural risk minimization	**TSVM**	Twin support vector machine
SVR	Support vector regression		

Chapter 1

Introduction

Machine learning aims to construct predictive rules from data. In supervised learning, the available information is a finite training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the ith input sample and y_i denotes either a class label or a real-valued response. The central objective is to learn a function $f : \mathbb{R}^d \to \mathcal{Y}$ that predicts accurately on unseen samples, not merely on the data used for training. This requirement makes supervised learning fundamentally a problem of generalization.

From the viewpoint of statistical learning theory, the performance of a prediction function is measured through the expected risk

$$R(f) = \mathbb{E}_{(\mathbf{x},y)} \left[\mathcal{L}\big(y, f(\mathbf{x})\big) \right], \tag{1.1}$$

where $\mathcal{L}(\cdot, \cdot)$ is a loss function that measures the discrepancy between the true output and the predicted output [1, 2]. Since the true data-generating distribution is unknown in practice, the expected risk cannot be evaluated directly. Learning algorithms therefore operate on the empirical risk

$$R_{\text{emp}}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\big(y_i, f(\mathbf{x}_i)\big). \tag{1.2}$$

To avoid overfitting, the empirical term is usually combined with a regularization term. A

broad family of supervised learning models can therefore be written as

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big) + \lambda \Omega(\theta), \tag{1.3}$$

where θ denotes the trainable parameters, $\Omega(\theta)$ is a regularizer, and $\lambda > 0$ controls the trade-off between data fitting and model complexity.

The formulation in (1.3) is compact, but it contains the main design decisions of supervised learning. The loss function decides how the model reacts to errors. The regularizer decides how complex the learned function may become. The representation f_θ decides what class of functions is available to the learner. The optimization method decides how the model is actually obtained. When data are clean, balanced, and generated under stable assumptions, many classical choices work well. In real applications, however, the situation is often less favorable. Labels may be corrupted, measurements may contain outliers, classes may be highly imbalanced, feature dimensions may be large, and numerical computations may be unstable. Therefore, the quality of learning depends not only on model expressiveness, but also on whether the loss, weighting scheme, representation, and optimization procedure remain reliable under imperfect conditions.

This thesis is motivated by this gap between idealized learning formulations and realistic data conditions. The title, *Robust and Efficient Machine Learning under Data Uncertainty: From Loss Functions to Data-Aware Representations*, reflects a progression from robust objective design to data-aware representation construction. First, noise and outliers can dominate the empirical risk by producing very large errors. Second, class imbalance can bias the learning objective toward the majority class and hide the minority class, even when minority-class predictions are practically important. Third, randomized neural networks provide computational efficiency by using randomly generated representations, but their reliability depends on three coupled issues: least-squares output learning may reduce robust-

ness under outliers, matrix inversion or pseudo-inversion may reduce numerical stability when the hidden matrix is ill-conditioned, and randomly generated hidden features may improve representation power while also introducing variability when they are poorly aligned with the data structure. These difficulties often occur together. For example, biomedical diagnosis may involve limited samples, noisy measurements, imbalanced classes, high-dimensional features, and severe consequences for minority-class errors.

The goal of this thesis is to develop learning methods that are both robust and efficient. Robustness means that the learned model should not be unduly controlled by corrupted observations, class skewness, or unstable random representations. Efficiency means that the method should remain computationally practical and should avoid unnecessary optimization burden. The thesis approaches this goal by redesigning four core components of learning: loss functions, sample weights, optimization strategies, and randomized representations.

Scope and Taxonomy of Uncertainty

In the established machine-learning taxonomy, *aleatoric uncertainty* denotes irreducible variability in observations or outcomes, whereas *epistemic uncertainty* denotes reducible uncertainty caused by limited knowledge about the data-generating process, model, or parameters [3]. These notions should be distinguished from other difficulties that can reduce predictive reliability. Label or measurement noise, local class overlap, and ambiguous observations are primarily data-related phenomena and may contribute to aleatoric uncertainty. Class imbalance is a distributional skew that changes the relative influence of classes in empirical risk; it is not, by itself, a type of uncertainty. Ill-conditioning is a numerical property of the learning problem, while variation caused by randomly initialized hidden features is an algorithmic or representation-level source of variability that can contribute to epistemic uncertainty about the learned predictor.

Accordingly, the phrase *data uncertainty* in the title is used as an operational umbrella for learning under imperfect, heterogeneous, or unreliable data conditions, not to assert that noise, imbalance, numerical instability, and initialization variability are equivalent forms of uncertainty. Chapters 3–5 address sensitivity to corrupted labels, outliers, and residual contamination through robust loss functions. Chapter 6 addresses class imbalance and local sample ambiguity through adaptive weighting. Chapter 7 addresses loss-induced sensitivity and numerical instability in randomized neural learning, while Chapter 8 addresses variability caused by data-agnostic random representations. The thesis therefore studies robustness to these distinct sources of unreliability; it does not claim to perform formal probabilistic uncertainty quantification.

1.1 Learning as an Interaction Between Loss, Weights, and Representation

Equation (1.3) shows that learning is not governed by the model family alone. The same decision function class can behave very differently under different losses or sample weights. To see this more clearly, consider the weighted regularized empirical risk

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mu_i \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big) + \lambda \Omega(\theta), \tag{1.4}$$

where $\mu_i \geq 0$ is the weight assigned to the ith sample. The expression in (1.4) contains three interacting mechanisms.

First, the loss $\mathcal{L}$ determines the cost of an error. If $\mathcal{L}$ is unbounded, samples with very large residuals or severe margin violations may dominate the objective. If $\mathcal{L}$ is bounded, the effect of extreme samples can be controlled. If $\mathcal{L}$ is smooth, gradient-based optimization can be applied directly without handling non-differentiable kink points in the objective.

If $\mathcal{L}$ has a zero-loss region, correctly classified or sufficiently accurate samples may stop contributing to the objective.

Second, the weight μ_i determines the importance of a sample before its error is evaluated. A minority-class sample, a boundary sample, a class-center sample, and an isolated suspected outlier need not contribute equally. Weighting is therefore a mechanism for injecting reliability, imbalance correction, and local uncertainty into the learning problem.

Third, the representation f_θ determines what the model can express. In hyperplane-based models, the representation is controlled by margins, kernels, and separating surfaces. In randomized neural networks, the representation is controlled by randomly generated hidden nodes, broad feature nodes, or stacked random layers. A powerful representation can improve prediction, but it can also amplify noise or create numerical instability if the optimization is not stable.

The thesis is built on the idea that robust learning should be studied through these interactions. A robust loss without a suitable model may be ineffective. A weighting scheme without a stable loss may still be dominated by outliers. A randomized representation trained through an outlier-sensitive or ill-conditioned output objective may remain computationally efficient, but its predictions can vary substantially under corrupted observations or unstable hidden matrices. The contributions of the thesis therefore redesign these components in a coordinated manner rather than treating robustness, imbalance, and randomness as isolated problems.

1.2 Role of Loss Functions in Learning

The loss function is the first and most direct way in which a learning algorithm defines what it means to make a mistake. In margin-based classification models, such as support vector machine (SVM) and twin support vector machine (TSVM), a loss translates the

signed margin $yf(\mathbf{x})$ or the corresponding margin-violation variable into a penalty. More generally, classification losses penalize disagreement between the true class label and the model score or probability estimate. In regression models, a loss translates the residual $y - f(\mathbf{x})$ into a penalty. Thus, the loss function determines which samples shape the learned model, how strongly they shape it, and whether extreme errors are allowed to dominate the training process.

1.2.1 Loss Functions as Influence Mechanisms

For a differentiable loss, the gradient of the empirical risk contains samplewise terms such as $\nabla_\theta \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big)$. Thus, the derivative of the loss determines the influence of the ith sample on the parameter update. If a loss has a derivative that grows without bound, then a single extreme sample may produce a disproportionately large update. If the derivative is bounded or decays for extreme errors, the sample influence is controlled. This is one reason why the design of robust losses is central to learning under outliers.

For example, the squared loss $\mathcal{L}(r) = r^2$, where $r = y - f(\mathbf{x})$, has derivative $2r$. A residual of magnitude 10 has ten times the derivative magnitude of a residual of magnitude 1. This behavior is useful when errors are generated by mild Gaussian-like noise, but it is undesirable when large residuals arise from outliers, incorrect labels, or measurement faults. The absolute loss $|r|$ controls the growth rate, but it is non-smooth at the origin. Huber loss [4] combines a quadratic region near zero with a linear region for large residuals, but it is still unbounded. These examples show that every loss imposes an assumption about how errors should affect learning.

The same issue appears in classification. Let $\gamma_i = y_i f(\mathbf{x}_i)$ be the signed classification margin, and let $u_i = 1 - \gamma_i$ be the corresponding margin-violation variable. A large positive γ_i indicates a correctly classified sample with high confidence, while a negative γ_i indicates

misclassification. A margin-based classification loss is often written as $\mathcal{L}(u_i)$, so that larger u_i corresponds to a more severe violation. Hinge loss, squared hinge loss, pinball loss, ramp loss, and truncated losses differ precisely in this influence pattern. Some are convex and easy to optimize, some are bounded and robust, some are smooth, and some are sparse. A central challenge is to combine these properties without making optimization impractical.

1.2.2 Classification Losses and Decision Boundaries

In hyperplane-based classification, the decision boundary is determined by samples near the margin and by samples that violate it. A typical margin-based objective can be written as

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\mathcal{L}\big(1 - y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b)\big). \tag{1.5}$$

The regularization term encourages a large-margin solution, while the loss term decides which samples resist that margin. If the loss grows indefinitely for samples with negative margins, then mislabeled samples or outliers can strongly distort the boundary. If the loss becomes flat for severe violations, then extreme samples have limited effect. If the loss is also smooth, the resulting optimization can be handled by gradient-based methods without abrupt changes in the objective.

This is why robust classification loss design is not a cosmetic modification. It changes the geometry of the learned classifier. Suppose a sample is far from its assigned class because its label is wrong. Under an unbounded loss, the optimization continues to spend effort trying to correct that sample, even though correcting it may damage the boundary for many clean samples. Under a bounded robust loss, the influence of such a sample is capped. The classifier can then focus on the reliable structure of the data rather than on extreme contradictions.

Another important property is sparsity. In SVM-type learning, samples that are correctly

classified with sufficient margin often have zero loss and do not influence the solution. This is valuable because it concentrates learning on informative samples near the decision boundary. However, sparsity alone is not enough. A loss may be sparse for well-classified samples but still unbounded for severely misclassified samples. The robust loss functions developed in this thesis aim to control both sides: they reduce unnecessary influence from already well-classified samples and suppress excessive influence from extreme outliers. This motivation leads to the Robust, Bounded, Sparse, and Smooth (RoBoSS) loss and the Wave loss developed in Chapters 3 and 4, where sparsity, boundedness, smoothness, and noise-insensitive margin behavior are studied in SVM and TSVM settings.

1.2.3 Regression Losses and Residual Control

In regression, the learning problem is shaped by residuals $r_i = y_i - f(\mathbf{x}_i)$. A typical regularized regression objective is

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(r_i) + \lambda \Omega(\theta). \tag{1.6}$$

The loss determines whether the model tries to fit every small fluctuation or ignores errors that are practically insignificant. This distinction is important because measurement noise is common in regression. A model that tries to fit every small deviation may overfit noise; a model that ignores all small deviations but reacts strongly to large residuals may still be sensitive to outliers.

Support vector regression (SVR) addresses the first issue through the ε-insensitive loss, which ignores residuals within an ε-tube [5]. This creates a tolerance zone and encourages sparsity. However, the classical ε-insensitive loss grows without bound outside the tube. Therefore, large outliers can still dominate the regression function. Robust regression requires a more careful structure: an insensitive zone for small errors, smooth behavior near

the transition region, and bounded influence for large residuals. This motivation leads to the HawkEye loss developed in Chapter 5.

1.2.4 Desirable Properties of Loss Functions

The preceding discussion shows that a robust and efficient loss should satisfy several properties:

1. **Boundedness:** The maximum penalty should be controlled so that extreme outliers cannot dominate the objective.

2. **Smoothness:** The loss should avoid abrupt changes when gradient-based or iterative optimization is used.

3. **Sparsity or insensitivity:** Samples that are already sufficiently well handled should not unnecessarily affect the model.

4. **Task alignment:** The loss should reflect the structure of the task, such as margin behavior in classification or residual tolerance in regression.

5. **Computational compatibility:** The loss should be integrable into existing learning frameworks without destroying their efficiency.

These properties are difficult to satisfy simultaneously. Bounded losses are often non-convex. Smooth losses may lose sparsity. Insensitive losses may be non-smooth at the boundary of the insensitive zone. This thesis develops loss functions that explicitly address these trade-offs: RoBoSS and Wave loss for robust classification, where RoBoSS provides sparsity and Wave provides asymmetric noise-insensitive behavior; HawkEye for bounded, smooth, and insensitive regression; and the fleXi Guardian (XG) loss for robust randomized neural learning.

1.3 Why Hyperplane-Based Models?

Hyperplane-based models are among the most important and interpretable families in supervised learning. Their basic decision function has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b, \tag{1.7}$$

where $\phi(\cdot)$ maps the input into a feature space, $\mathbf{w}$ is the normal vector of the hyperplane, and b is the bias. In binary classification, the sign of $f(\mathbf{x})$ determines the predicted class, while the magnitude $|f(\mathbf{x})|$ is related to the distance from the decision boundary. The separating surface is therefore geometrically meaningful.

1.3.1 Margin Principle and Generalization

SVMs provide the canonical example of hyperplane-based learning [1, 6]. For labels $y_i \in \{-1, +1\}$, the functional margin of a sample is $y_i f(\mathbf{x}_i) = y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)$. The soft-margin SVM objective can be written as

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m} \max\big(0, 1 - y_i f(\mathbf{x}_i)\big). \tag{1.8}$$

The first term encourages a large geometric margin, while the second term penalizes margin violations. The margin principle is theoretically important because controlling $\|\mathbf{w}\|_2$ restricts the capacity of the classifier and supports generalization [7]. It is also practically useful because the solution is determined by a subset of samples, namely the support vectors.

This structure makes SVM an ideal setting for robust loss design. The loss function directly determines which samples become influential. A noisy sample near the boundary may be genuinely informative, but a mislabeled sample far from its assigned class may be

harmful. A robust SVM loss should preserve the contribution of useful margin samples while suppressing the excessive effect of extreme corrupted samples. This balance is the central motivation behind the RoBoSS-SVM and Wave-SVM formulations developed in Chapters 3 and 4.

1.3.2 Least-Squares SVM and the Efficiency-Robustness Trade-Off

Least-squares support vector machine (LSSVM) replaces inequality constraints with equality constraints and uses a squared-error type objective [8, 9]. This leads to a system of linear equations rather than a quadratic programming problem. The computational convenience of LSSVM makes it attractive, especially when repeated training is required.

However, the least-squares structure also introduces sensitivity to outliers. Since squared error grows quadratically, large deviations receive very large penalties. In classification, mislabeled points can distort the decision boundary; in regression, abnormal residuals can pull the function away from the main data trend. Therefore, LSSVM-type models highlight a recurring theme of this thesis: computational efficiency must be balanced with robustness. The HawkEye and RAPID-based developments in Chapters 5 and 6 use this perspective by retaining the efficiency of least-squares frameworks while reducing sensitivity to unreliable samples.

1.3.3 Twin SVM and Class-Specific Geometry

TSVM modifies the SVM geometry by learning two nonparallel hyperplanes instead of one [10]. One hyperplane is fitted close to one class and away from the other, while the second hyperplane is fitted in the opposite way. A test sample is classified by comparing its distances from the two hyperplanes. This class-specific geometry can be computationally efficient because two smaller problems are solved, and it can be flexible when class

distributions are not well separated by one global boundary.

TSVM is particularly relevant to this thesis because class-specific hyperplanes can be useful for heterogeneous or imbalanced data. However, TSVM also depends heavily on the loss or constraints used to separate the classes. If the opposing class contains noisy or mislabeled samples, those samples may affect the construction of the class-specific hyperplane. Robust TSVM therefore requires robust loss design in a twin-hyperplane setting. This motivates the RoBoSS-based Twin Support Vector Machine (RoBoTS) and Wave-TSVM formulations in Chapters 3 and 4.

1.3.4 SVR and Robust Regression

SVR extends the margin idea to real-valued prediction [5]. Instead of constructing a separating margin between classes, SVR constructs an ε-insensitive tube around the regression function. Residuals inside the tube are ignored, while residuals outside the tube are penalized. This is useful because small deviations are often not meaningful in noisy measurements.

Nevertheless, classical SVR losses remain unbounded outside the insensitive tube. Therefore, the model may still be affected by large residuals caused by outliers. A robust regression loss should preserve the idea of an insensitive region while bounding the influence of extreme residuals and maintaining smooth optimization. This is the motivation behind the HawkEye loss developed in Chapter 5.

1.3.5 Why Hyperplane Models Remain Central

The hyperplane-based models considered in this thesis are not chosen merely because they are classical. They are chosen because they provide a mathematically tractable and geometrically transparent framework for studying robustness. Their geometry is clear, their

optimization objectives are explicit, and their sensitivity to loss functions and weights can be mathematically analyzed. In noisy or imbalanced settings, the learned hyperplane can shift due to a small number of unreliable samples. Therefore, improving loss functions and weighting schemes inside hyperplane-based models directly addresses a fundamental weakness while preserving a strong theoretical foundation.

1.4 Effect of Weighting Schemes in Learning

Loss functions decide how errors are penalized, but weighting schemes decide how much each sample matters before the error is evaluated. In real datasets, not all samples are equally reliable or equally important. A clean boundary sample may be valuable, a mislabeled sample may be harmful, a minority-class sample may need additional emphasis, and a sample in an overlapping region may require careful treatment. Weighting schemes provide a principled way to encode these differences.

1.4.1 Weighted Empirical Risk

The weighted empirical risk in (1.4) modifies the standard empirical objective by introducing μ_i. If $\mu_i = 1$ for all i, all samples contribute equally. If μ_i is small for a suspected outlier, that sample has reduced influence. If μ_i is larger for a minority-class sample, the learning process can compensate for class skewness.

This mechanism is simple but powerful. In gradient-based learning, the contribution of the ith sample to the gradient is multiplied by μ_i. In least-squares models, the sample weights alter the normal equations and therefore change the fitted function. In SVM-type models, weights affect the penalty assigned to margin violations. Thus, sample weighting changes the effective distribution seen by the learner.

1.4.2 Class Imbalance

Class imbalance occurs when one class contains far more samples than another. If the empirical risk is minimized uniformly, the majority class contributes more terms to the objective. A classifier can therefore achieve high overall accuracy by performing well on the majority class while failing on the minority class. This is particularly problematic in medical diagnosis, fraud detection, fault detection, and other applications where minority samples may represent rare but critical events.

Data-level methods such as oversampling and undersampling attempt to change the training distribution [11]. However, oversampling can introduce redundancy, and undersampling can discard useful majority-class information. Algorithm-level weighting avoids changing the data directly; instead, it changes the influence of samples in the objective. This makes weighting attractive for models such as LSSVM, where sample weights can be incorporated into the training formulation.

1.4.3 Uncertainty and Local Reliability

Imbalance is not the only reason to weight samples. A sample may be uncertain because it lies near a decision boundary, because it is surrounded by samples from other classes, or because it is far from the main body of its own class. Fuzzy SVM methods use membership values to reduce the influence of outliers and unreliable samples [12, 13]. Many such schemes use distance from a class center or distance from an initial hyperplane.

Distance-based weighting is useful, but it may not be sufficient. A sample close to a class center may still lie in a region of class overlap. A sample far from the center may be a legitimate minority pattern rather than an outlier. A majority-class sample and a minority-class sample with the same distance profile may require different treatment because the class priors are different. Therefore, robust weighting should combine multiple signals:

global imbalance, class-center proximity, and local class probability. This is the motivation behind the Radius-Adaptive, Probability-Aware, and Imbalance-Driven (RAPID) weighting scheme in Chapter 6.

1.5 Randomized Neural Networks as Efficient Alternatives to Fully Trained Neural Networks

Artificial neural networks are powerful function approximators. Deep learning models, in particular, have achieved strong performance in image analysis, speech processing, natural language processing, and many other domains [14, 15]. Their strength comes from learning multiple layers of nonlinear transformations. However, this strength comes with computational cost. Standard neural networks usually train a large number of weights by backpropagation, requiring iterative optimization, careful learning-rate selection, initialization choices, and often substantial computational resources.

For many tabular, biomedical, and medium-scale datasets, fully trained deep architectures may be unnecessarily expensive or unstable. The number of available samples may be limited, and the cost of tuning a deep model may not be justified. This motivates alternative neural models that retain nonlinear representation power while simplifying training. Randomized neural networks provide one such alternative.

1.5.1 Basic Idea of Randomized Neural Networks

Randomized neural networks reduce training cost by randomly generating part of the network and learning only the output layer. For an input matrix $X \in \mathbb{R}^{m\times d}$, a typical hidden representation is

$$H = g(XW + \mathbf{1}\mathbf{b}^\top), \tag{1.9}$$

where W is a randomly sampled input-to-hidden weight matrix, $\mathbf{b}$ is a random bias vector, and $g(\cdot)$ is a nonlinear activation function. Let Z denote the matrix supplied to the trainable output layer. For a basic extreme learning machine (ELM)-type model, $Z = H$; for a random vector functional link (RVFL)-type model with direct links, $Z = [X, H]$; and for broader randomized architectures, Z may collect feature and enhancement nodes. The output layer is then commonly trained by solving a regularized least-squares problem:

$$\min_{\Theta} \ \|Z\Theta - Y\|_F^2 + \lambda\|\Theta\|_F^2. \tag{1.10}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, i.e., $\|A\|_F = \left(\sum_{i,j} A_{ij}^2\right)^{1/2}$. A common closed-form solution is

$$\Theta = (Z^\top Z + \lambda I)^{-1} Z^\top Y. \tag{1.11}$$

This formulation explains why randomized neural networks are efficient. The nonlinear representation is produced without iterative hidden-layer training, and the output weights are computed through a linear algebraic solution.

1.5.2 Random Vector Functional Link, Extreme Learning Machine, Broad Learning System, and Deep RVFL

RVFL networks use random hidden nodes together with direct input-to-output connections [16]. The direct links allow the output layer to use both the original input and the random nonlinear features. ELMs are closely related single-hidden-layer randomized networks without direct links [17]. Both models are attractive because they can train quickly while still providing nonlinear mapping capability.

The broad learning system (BLS) extends the randomized philosophy in a flat architecture [18, 19]. Instead of going deep, BLS constructs feature mapping nodes and en-

hancement nodes in a broad structure and learns the output weights efficiently. Deep RVFL extends RVFL by stacking multiple randomized hidden layers while preserving direct links from earlier representations to the output layer [20]. Each layer produces a new nonlinear transformation from fixed random weights, and the output layer can use a concatenation of shallow and deep randomized features. This design increases representation capacity without full backpropagation through all hidden weights, but it also makes the quality, diversity, and stability of random features more important.

These models are important in this thesis because they provide a natural setting for studying efficient learning under uncertainty. They are fast, flexible, and suitable for repeated experiments. At the same time, they are sensitive to the quality of random representations, the conditioning of the hidden matrix, and the choice of training loss.

1.5.3 Benefits Over Fully Trained Neural Networks

Randomized neural networks have several practical benefits over fully trained neural networks:

1. **Reduced training cost:** Hidden-layer parameters are not updated iteratively, which reduces computational burden.

2. **Simpler optimization:** Output-layer learning is often reduced to ridge regression or a related convex problem.

3. **Fast experimentation:** Since training is inexpensive, randomized models are useful when many datasets, noise levels, or parameter settings must be evaluated.

4. **Suitability for compact tabular problems:** Their low training cost and moderate model size can be attractive for small- and medium-sized tabular datasets. In this thesis, the biomedical applicability of randomized neural networks is demonstrated in

Chapter 8; the robust and inversion-free formulations in Chapter 7 are evaluated on generic UCI and KEEL benchmarks rather than on a separate biomedical case study.

5. **Extensibility:** Randomized networks can be combined with robust losses, fuzzy weights, ensemble strategies, broad architectures, and dependency-aware initialization.

These benefits explain why randomized neural networks are not merely simplified neural networks. They are a distinct modeling philosophy: use random nonlinear transformations to generate a rich representation, then train a stable and efficient readout.

1.5.4 Limitations of Randomized Neural Networks

The efficiency of randomized neural networks comes with important limitations. First, the closed-form solution in (1.11) requires matrix inversion or pseudo-inversion. If $Z^\top Z + \lambda I$ is ill-conditioned, small perturbations in Z or Y can produce large changes in Θ. This is particularly concerning when data are noisy, the number of hidden nodes is large, or hidden features are highly correlated.

Second, many randomized neural networks use squared loss in their output layer. As discussed earlier, squared loss is sensitive to outliers because large residuals receive quadratically large penalties. Therefore, even if the architecture is efficient, the learning objective may be fragile under corrupted observations. This motivates the robust loss-based and inversion-free randomized neural learning strategy developed in Chapter 7.

Third, the random weights are often sampled independently from simple distributions. This ignores the dependence structure among input features. In real datasets, features are rarely independent. If random weights are generated without considering feature dependence, the hidden representation may be poorly aligned with the geometry of the data. This motivates the copula-based initialization strategy developed in Chapter 8.

Thus, randomized neural networks provide efficiency, but robustness is not automatic. The thesis addresses this by developing robust and inversion-free randomized neural learning frameworks and dependency-aware random initialization.

1.6 Noise, Imbalance, and Randomness as a Unified Problem

The methods considered in this thesis may appear different at first glance. SVM and TSVM are hyperplane-based classifiers; SVR and least-squares support vector regression (LSSVR) are regression models; RVFL, deep RVFL, and BLS are randomized neural models. However, they share a common learning structure. Their performance depends on the loss function, the influence assigned to each sample, the representation used by the model, and the numerical stability of the optimization procedure.

Noise primarily attacks the loss mechanism. A mislabeled or outlying sample can produce a large margin violation or residual. If the loss is unbounded, that sample can dominate the empirical risk. Imbalance attacks the sample influence mechanism. Majority-class samples appear more frequently and can control the learned decision rule. Randomness attacks the representation and optimization mechanism. Random hidden features may be unstable, and matrix inversion may amplify perturbations.

In practice, these effects interact. A noisy minority-class sample may be incorrectly treated as an outlier. A boundary sample may be down-weighted even though it is informative. A randomized representation may make class overlap worse if the random features do not align with the input structure. A least-squares solution may fit outliers aggressively because of squared loss. Therefore, robust learning requires coordinated design across losses, weights, representations, and optimization.

This thesis develops such a coordinated approach. Chapters 3 and 4 design robust classification losses and integrate them into SVM and TSVM. Chapter 5 designs a robust regression loss and integrates it into LSSVR. Chapter 6 develops a weighting scheme that accounts for imbalance and uncertainty. Chapter 7 extends robust loss design to randomized neural networks while avoiding unstable inversion. Chapter 8 addresses randomness itself by aligning random weight initialization with feature dependence through copula theory [21, 22].

1.7 Research Gaps

The preceding discussion leads to several research gaps that define the direction of this thesis.

1.7.1 Gap 1: Robust Classification Losses Need a Better Balance of Boundedness, Smoothness, Sparsity, and Asymmetry

Classical margin losses are useful but incomplete for noisy classification. Convex losses are often easy to optimize, but many are unbounded and can be sensitive to outliers. Bounded losses reduce outlier influence, but they may be non-smooth or difficult to optimize. Sparse losses reduce the effect of well-classified samples, but they may still overpenalize severely corrupted samples. In addition, asymmetric margin behavior can be useful under label noise because it allows the loss to treat correctly classified and incorrectly classified regions differently; in the Wave loss, this asymmetry lets correctly classified samples retain a controlled penalty contribution instead of forcing a symmetric response around the margin. A robust classification loss should balance these properties in a single formulation. This motivates the RoBoSS and Wave loss functions developed in Chapters 3 and 4.

1.7.2 Gap 2: Robust Regression Requires Boundedness Together with an Insensitive Zone

Regression data often contain small measurement noise and occasional large outliers. A useful regression loss should not overreact to small residuals, but it should also prevent large residuals from dominating the model. Classical ε-insensitive loss provides tolerance but is unbounded; bounded losses control outliers but may lack an insensitive zone or smooth behavior. This motivates the HawkEye loss developed in Chapter 5.

1.7.3 Gap 3: Class Imbalance and Sample Uncertainty Are Often Treated Separately

Many methods handle class imbalance using global class ratios, while many fuzzy methods handle uncertainty using distance-based membership values. In real datasets, these two issues interact. A sample may be uncertain because of local class overlap, because it lies far from the class center, or because it belongs to the minority class. A weighting scheme should therefore combine local probability, geometric proximity, and imbalance correction. This motivates the RAPID weighting scheme developed in Chapter 6.

1.7.4 Gap 4: Randomized Neural Networks Need Robust and Inversion-Free Learning

Randomized neural networks are computationally efficient, but direct matrix inversion can be unstable under noisy or ill-conditioned hidden representations. Moreover, squared-loss output learning can be sensitive to outliers. Robust randomized learning should therefore combine robust losses with optimization procedures that avoid unstable inversion. This motivates the fleXi Guardian random vector functional link (XG-RVFL) and Wave-BLS frameworks developed in Chapter 7.

1.7.5 Gap 5: Independent Random Initialization Ignores Feature Dependence

The random hidden parameters in RVFL, ELM, deep RVFL, and BLS models are commonly sampled independently. This ignores dependence among input features. Since real datasets often contain correlated or dependent variables, independent random initialization may generate representations that are not well aligned with the data. Copula theory provides a rigorous framework for modeling dependence separately from marginal distributions. This motivates the Copula-Aligned Weight Initialization (CAWI) framework developed in Chapter 8.

1.8 Scope of the Thesis

The thesis focuses on supervised classification and regression. The classification part considers SVM, TSVM, LSSVM, RVFL, deep RVFL, and BLS frameworks. The regression part considers SVR and LSSVR. The thesis does not aim to replace these models with a single universal architecture. Instead, it develops robust and efficient components that can be integrated into established learning frameworks.

The scope can be summarized through four technical directions:

1. **Robust loss design for classification:** bounded, smooth, sparse, and asymmetric loss structures are developed to reduce the influence of outliers and label noise while retaining useful margin behavior.

2. **Robust loss design for regression:** a bounded, smooth, and insensitive loss is developed to tolerate small errors and control large residuals.

3. **Adaptive weighting for imbalanced and uncertain data:** sample weights are de-

signed using class-center proximity, local class probability, and imbalance information.

4. **Stable randomized neural learning:** robust losses, inversion-free optimization, and copula-based dependency-aware initialization are used to improve randomized neural networks.

Together, these directions form a coherent research program: redesign the objective, sample influence, optimization strategy, and randomized representation so that learning becomes less sensitive to corrupted samples, skewed class distributions, ill-conditioned inversions, and arbitrary random initialization.

1.9 Research Objectives

The main objective of this thesis is to develop robust and efficient supervised learning methods that can operate reliably under noise, class imbalance, and randomness. More specifically, the thesis pursues the following objectives:

1. To design bounded, smooth, sparse, and asymmetric loss functions for classification that improve robustness, noise insensitivity, and optimization compatibility.

2. To incorporate robust loss functions into SVM and TSVM models and develop efficient optimization procedures for the resulting non-convex objectives.

3. To construct a robust regression loss function with an insensitive zone and integrate it into an LSSVR framework.

4. To develop adaptive sample-weighting strategies for class-imbalanced and noisy datasets, especially in biomedical diagnosis.

5. To improve randomized neural networks by replacing unstable closed-form inversion with gradient-based or inversion-free learning strategies.

6. To design a dependency-aware initialization mechanism for randomized neural networks using copula theory.

7. To validate the developed methods through extensive experiments on benchmark, noisy, imbalanced, and biomedical datasets using suitable evaluation metrics and statistical tests.

1.10 Major Contributions

The major contributions of this thesis are organized as follows.

1. **RoBoSS loss for robust classification:** A robust, bounded, sparse, and smooth loss function is developed for supervised classification. The loss is integrated into SVM and then extended to twin SVM through RoBoTS and inverse-free RoBoTS formulations. This contribution addresses the difficulty of designing a loss that is robust to outliers, sparse in its support-vector behavior, and smooth enough for efficient optimization.

2. **Wave loss for smooth robust SVM and TSVM learning:** A bounded, smooth, and asymmetric classification loss is developed and embedded into SVM and TSVM frameworks. The asymmetry of the loss allows correctly classified samples to retain a controlled influence, which improves noise-insensitive learning while preserving optimization-friendly differentiability.

3. **HawkEye loss for robust regression:** The HawkEye regression loss combines boundedness, smoothness, and an insensitive zone. By incorporating HawkEye into

LSSVR, the thesis develops HawkEye least-squares support vector regression (HE-LSSVR), which is optimized using an adaptive moment estimation strategy in an inverse-free manner [23].

4. **RAPID weighting scheme for class-imbalanced learning:** A radius-adaptive, probability-aware, and class-imbalance-driven weighting scheme is developed for class-imbalanced learning under uncertainty. RAPID combines proximity to class centers, local class-probability information, and imbalance-ratio correction, and is integrated into LSSVM for robust classification.

5. **Robust and inversion-free randomized neural networks:** The thesis extends robust loss design to randomized neural models. It develops the XG-RVFL framework and Wave-BLS framework to improve learning under noisy and uncertain data while avoiding instability associated with direct matrix inversion.

6. **Copula-aligned weight initialization:** A dependency-aware initialization strategy is developed for randomized neural networks. Instead of sampling random weights independently, CAWI uses copula theory to align the dependence structure of the random weights with that of the input features, improving stability and performance.

7. **Empirical and statistical validation:** The developed models are evaluated on benchmark datasets, noisy datasets, imbalanced datasets, and biomedical datasets. The empirical comparisons are supported using standard performance metrics and statistical tests, including Friedman, Nemenyi, Wilcoxon, and win-tie-loss analyses where appropriate [24, 25].

1.11 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature Review*, establishes the mathematical and methodological background used throughout the thesis. It discusses regularized empirical risk, margin-based learning, support vector machine (SVM), twin support vector machine (TSVM), support vector regression (SVR), classification and regression losses, weighting schemes for class imbalance and uncertainty, randomized neural networks, copula theory, evaluation metrics, and statistical comparison tests. This chapter fixes the notation and conceptual foundation needed for the later technical chapters.

Chapter 3, *RoBoSS: A Robust, Bounded, Sparse, and Smooth Loss Function for Supervised Learning*, develops the RoBoSS loss as a bounded, sparse, smooth, and robust classification loss. The chapter first integrates RoBoSS into the SVM framework and analyzes its classification-calibrated and generalization properties. It then extends the same loss principle to TSVM by developing RoBoSS-based Twin Support Vector Machine (RoBoTS) and inverse-free RoBoTS. The chapter demonstrates how robust loss design improves classification under clean, noisy, outlier-contaminated, and biomedical settings.

Chapter 4, *Advancing Supervised Learning with the Wave Loss Function: A Robust and Smooth Approach*, introduces the Wave loss function for robust supervised classification. The chapter studies a smooth, bounded, asymmetric loss that is designed to reduce the influence of outliers and noisy samples while preserving optimization-friendly differentiability. It develops Wave-SVM and Wave-TSVM by embedding the Wave loss into least-squares SVM and TSVM formulations, and evaluates their effectiveness on benchmark, noisy, and biomedical datasets.

Chapter 5, *HawkEye: A Robust Loss Function for Regression with Bounded, Smooth,*

and Insensitive Zone Characteristics, develops the HawkEye loss for robust regression. The chapter addresses the gap between non-smooth bounded regression losses and smooth bounded losses that lack an insensitive zone. By combining boundedness, smoothness, and an insensitive region, HawkEye is integrated into least-squares SVR to obtain HE-LSSVR, which is optimized through an inverse-free Adam-based approach and evaluated on synthetic, benchmark, time-series, and brain-age prediction tasks.

Chapter 6, *Dual-Center RAPID-LSSVM: Radius-Adaptive, Probability, and Class-Imbalance-Driven Weighting for Alzheimer's Diagnosis*, develops an adaptive weighting framework for class-imbalanced and uncertain data. The chapter introduces RAPID, which combines radius-adaptive proximity, local class-probability information, and class-imbalance correction. By embedding these weights into LSSVM, the chapter develops RAPID-LSSVM-I and RAPID-LSSVM-II using mean-center and median-center designs, and studies their performance on benchmark datasets and Alzheimer's disease diagnosis.

Chapter 7, *Robust and Inversion-Free Learning in Randomized Neural Networks*, extends robust loss-function design to randomized neural networks. The chapter first develops XG-RVFL by introducing the fleXi Guardian loss and an inversion-free optimization strategy for binary and multiclass classification. It then develops Wave-BLS by integrating the Wave loss into broad learning systems and solving the resulting objective without matrix inversion. Together, these models address noise sensitivity, outlier sensitivity, and numerical instability in randomized learning.

Chapter 8, *CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks*, develops a dependency-aware initialization strategy for randomized neural networks. Instead of sampling input-to-hidden weights independently, CAWI fits copulas to the empirical dependence structure of the input features and samples frozen random weights from the learned dependence model. The chapter shows how this copula-aligned initialization

improves randomized representations while preserving closed-form output learning.

Chapter 9, *Conclusions and Future Directions*, concludes the thesis by synthesizing the major findings and outlining future research directions for robust losses, adaptive weighting, multiclass extensions, deep randomized neural networks, and automatic copula selection.

Chapter 2

Literature Review

This chapter formalizes the methodological foundations motivated in Chapter 1. The introduction explains why robustness, weighting, representation, and numerical stability must be studied together when learning from noisy, imbalanced, and uncertain data. The present chapter develops the corresponding technical background in a systematic form: notation and regularized empirical risk, margin-based learning, classification and regression losses, sample weighting under uncertainty, randomized neural networks, copula-based dependency modeling, optimization stability, evaluation metrics, and statistical comparison tests.

2.1 Notation and Problem Setting

Let the training set be denoted by $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the ith input pattern, y_i is its class label for classification or real-valued response for regression, m is the number of samples, and d is the number of input features. In binary classification, labels are encoded as $y_i \in \{-1, +1\}$. For multiclass classification with c classes, the label matrix is denoted by $Y \in \mathbb{R}^{m \times c}$, usually using one-hot encoding. The input matrix is denoted by $X = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_m^\top]^\top \in \mathbb{R}^{m \times d}$.

For a linear predictor or linear separating hyperplane used in margin-based models, the

decision function is written as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and $b \in \mathbb{R}$ is the bias. In kernel-based models, the input is mapped to a reproducing kernel Hilbert space through $\phi(\mathbf{x})$, and the kernel is written as $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. For randomized neural networks, the hidden-layer output matrix is denoted by H, the input-to-hidden random weight matrix by W, the random hidden bias by $\mathbf{b}$, and the trainable output weight matrix by Θ. Throughout the thesis, the signed classification margin is denoted by $\gamma_i = y_i f(\mathbf{x}_i)$, and the corresponding margin-violation variable used as the argument of many classification losses is denoted by $u_i = 1 - \gamma_i = 1 - y_i f(\mathbf{x}_i)$. The regression residual is denoted by $r_i = y_i - f(\mathbf{x}_i)$. Unless explicitly redefined in a chapter-specific model, m denotes the number of samples, d denotes the number of input features, and c denotes the number of classes or output coordinates. Some model-specific derivations use conventional local symbols such as l, n, N, h, k, a, λ, η, γ, δ, or ε; whenever such reuse is unavoidable, the symbol is defined locally and its meaning is restricted to that context.

The common learning objective underlying the models studied in this thesis can be viewed as regularized empirical risk minimization:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big) + \lambda \Omega(\theta), \tag{2.1}$$

where $\mathcal{L}(\cdot, \cdot)$ is a task-specific loss function, $\Omega(\theta)$ is a regularizer, and $\lambda > 0$ controls the trade-off between data fitting and model complexity. Later chapters use this framework by focusing on different components: the loss function, the sample weights, the optimization strategy, and the randomized representation. The core symbols used throughout the chapter and the later thesis are summarized in Table 2.1.

Table 2.1: Core notation used throughout the thesis.

Symbol	Meaning
m, d, c	Number of samples, input features, and classes, respectively
$\mathbf{x}_i, y_i$	Input vector and label/response of the ith sample
X, Y	Input matrix and target matrix
$\mathbf{w}, b$	Weight vector and bias of a linear model
$f(\mathbf{x})$	Prediction or decision score of a model
$\gamma_i = y_i f(\mathbf{x}_i)$	Signed classification margin
$u_i = 1 - y_i f(\mathbf{x}_i)$	Margin-violation variable used in classification losses
$r_i = y_i - f(\mathbf{x}_i)$	Regression residual
$\phi(\cdot), K(\cdot, \cdot)$	Feature map and kernel function
$H, W, \mathbf{b}$	Hidden output matrix, random weights, and random bias in RdNNs
Θ	Trainable output weight matrix in randomized neural networks
q, p, s, r	BLS feature windows, feature nodes, enhancement windows, and enhancement nodes
$\mathcal{L}$	Loss function
λ, C	Regularization or penalty parameters
μ_i	Sample membership or weight assigned to the ith sample

2.2 Margin-Based Learning

Margin-based learning is a central foundation of this thesis. Support vector machines (SVMs), least-squares SVMs (LSSVMs), twin SVMs (TSVMs), and related robust variants seek separating hyperplanes with strong generalization ability by controlling geometric margin and empirical error. They are therefore natural models for studying how loss design and sample weighting influence the learned decision surface.

2.2.1 Support Vector Machines

The support vector machine introduced by Cortes and Vapnik [6] and grounded in statistical learning theory [1] constructs a maximum-margin separating hyperplane. For binary classification with $y_i \in \{-1, +1\}$, the soft-margin primal SVM is

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m} \xi_i, \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{2.2}$$

where $C > 0$ is the penalty parameter and ξ_i are slack variables. Equivalently, the empirical component of the SVM objective uses the hinge loss

$$\mathcal{L}_{\text{hinge}}(\gamma) = \max(0, 1-\gamma) = \max(0, u), \qquad \gamma = yf(\mathbf{x}), \quad u = 1-\gamma. \tag{2.3}$$

Thus, correctly classified samples outside the unit margin have zero loss, samples inside the margin have positive loss, and misclassified samples receive a penalty that grows linearly with the margin violation. Figure 2.1 illustrates the separating hyperplane and the associated unit-margin band.

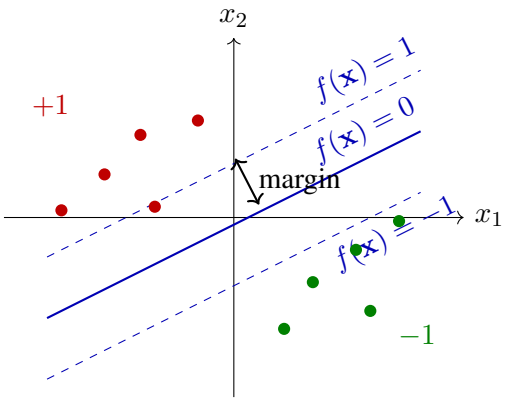


Figure 2.1: Geometric view of SVM. The central hyperplane separates the two classes, while the two dashed hyperplanes define the unit-margin band. Samples on or inside this band determine the solution and become support vectors.

The strengths of SVM include convexity, a clear geometric interpretation, and kernelization. However, classical SVMs can be sensitive to noisy labels and outliers because unbounded or linearly increasing losses can assign large influence to atypical observations. This limitation motivates the broader study of boundedness, smoothness, sparsity, and robustness in classification loss design.

2.2.2 Least-Squares SVM

Least-squares SVM (LSSVM) replaces inequality constraints in SVM with equality constraints and uses a squared-error criterion [8]. A typical binary LSSVM formulation is

$$\begin{aligned} \min_{\mathbf{w},b,\mathbf{e}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{m} e_i^2, \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) = 1 - e_i, \quad i = 1, \ldots, m. \end{aligned} \tag{2.4}$$

The equality constraints lead to a system of linear equations rather than a quadratic programming problem, which makes LSSVM computationally attractive. At the same time, the squared-error term is highly sensitive to outliers because large residuals receive quadratically large penalties. This sensitivity is central to the robust regression and robust classification developments in this thesis.

2.2.3 Twin Support Vector Machines

Twin support vector machines construct two nonparallel hyperplanes, each of which is close to one class and far from the other [10]. Let $A \in \mathbb{R}^{m_1 \times d}$ contain positive-class samples and $B \in \mathbb{R}^{m_2 \times d}$ contain negative-class samples. TSVM learns two hyperplanes $\mathbf{w}_1^\top \mathbf{x} + b_1 = 0$ and $\mathbf{w}_2^\top \mathbf{x} + b_2 = 0$. A common linear TSVM formulation is

$$\begin{aligned} \min_{\mathbf{w}_1,b_1,\boldsymbol{\xi}} \quad & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|_2^2 + C_1 \mathbf{e}_2^\top \boldsymbol{\xi}, \\ \text{s.t.} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) + \boldsymbol{\xi} \geq \mathbf{e}_2, \quad \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{2.5}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2,b_2,\boldsymbol{\eta}} \quad & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|_2^2 + C_2 \mathbf{e}_1^\top \boldsymbol{\eta}, \\ \text{s.t.} \quad & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) + \boldsymbol{\eta} \geq \mathbf{e}_1, \quad \boldsymbol{\eta} \geq \mathbf{0}, \end{aligned} \tag{2.6}$$

where $\mathbf{e}_1$ and $\mathbf{e}_2$ are all-one vectors of compatible dimensions. The first problem fits a hyperplane near A and pushes it away from B, whereas the second problem fits a hyperplane

near B and pushes it away from A. A test sample $\mathbf{x}$ is assigned by the normalized distance rule

$$\text{class}(\mathbf{x}) = \arg\min_{j\in\{1,2\}} \frac{|\mathbf{w}_j^\top \mathbf{x} + b_j|}{\|\mathbf{w}_j\|_2}. \tag{2.7}$$

The two-hyperplane geometry and nearest-hyperplane decision rule are shown in Figure 2.2.

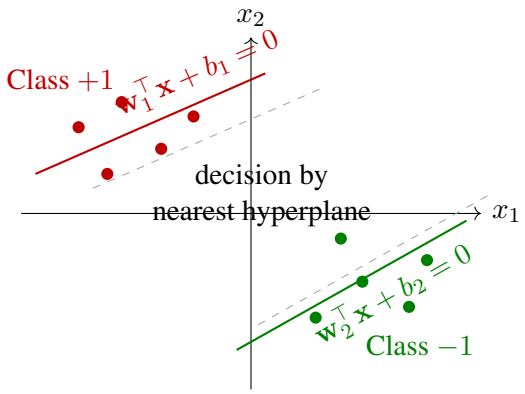


Figure 2.2: Geometric view of TSVM. Instead of a single separating hyperplane, TSVM learns two nonparallel class-specific hyperplanes and classifies a test sample using its relative distance to these hyperplanes.

TSVM often reduces computational cost compared with standard SVM because it solves two smaller optimization problems instead of one larger problem. Its class-specific geometry is also attractive for imbalanced or heterogeneous data. However, like other margin-based models, TSVM can inherit sensitivity to noise, outliers, and class overlap when its loss or constraints are not robust.

2.2.4 Support Vector Regression

Support vector regression (SVR) extends the maximum-margin idea of SVM to real-valued prediction [5]. Given regression samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, the SVR function is $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$. Instead of requiring exact interpolation, SVR introduces an ε-insensitive tube around the target response. Errors inside the tube are ignored, while deviations outside the

tube are penalized. The soft-margin SVR primal problem is

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*), \\ \text{s.t.} \quad & y_i - \mathbf{w}^\top\phi(\mathbf{x}_i) - b \le \varepsilon + \xi_i, \\ & \mathbf{w}^\top\phi(\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \ge 0, \quad i = 1, \ldots, m. \end{aligned} \tag{2.8}$$

Equivalently, SVR minimizes a regularized empirical risk using the ε-insensitive loss $\mathcal{L}_\varepsilon(r) = \max(0, |r| - \varepsilon)$, where $r = y - f(\mathbf{x})$. The insensitive tube gives SVR sparsity because samples inside the tube have zero loss and do not become support vectors. Figure 2.3 shows this ε-tube geometry. However, the classical ε-insensitive loss grows without bound outside the tube, so large outliers can still strongly affect the learned function. This limitation motivates robust regression losses that combine insensitive regions, bounded influence, and smooth optimization behavior.

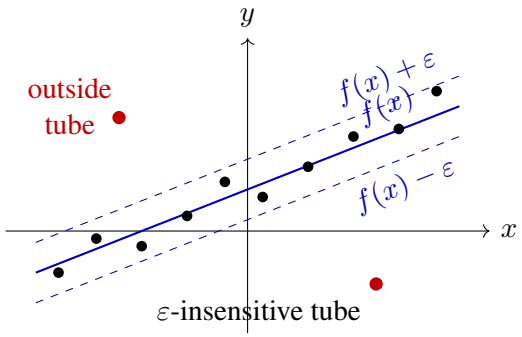


Figure 2.3: Geometric interpretation of SVR. Samples inside the ε-tube incur no loss, while samples outside the tube are penalized through slack variables.

2.3 Classification Loss Functions

The design of the classification loss determines how a classifier responds to correctly classified samples, near-boundary samples, mislabeled samples, and severe outliers. In

margin-based classification, it is convenient to define the margin violation as $u = 1 - yf(\mathbf{x})$. Thus $u \leq 0$ indicates that the sample is correctly classified with at least unit margin, $0 < u < 1$ represents a margin violation, and larger u indicates a more serious error. A robust classification loss should ideally balance sparsity, bounded influence, smooth optimization behavior, and classification calibration [26]. Classical losses typically satisfy only part of this set of requirements.

Table 2.2 summarizes the representative classification losses used as baselines or methodological reference points in later chapters. The subsequent subsections discuss these losses in terms of convexity, smoothness, boundedness, sparsity, and robustness.

2.3.1 Convex but Non-Smooth Losses: Hinge and Pinball

The hinge loss used in classical SVM is

$$\mathfrak{L}_{\text{hinge}}(u) = \begin{cases} u, & u > 0, \\ 0, & u \leq 0. \end{cases} \tag{2.9}$$

It is convex and sparse, because samples satisfying the margin constraint contribute zero loss. Its limitations are non-smoothness at $u = 0$ and unbounded linear growth for large violations, which allows mislabeled or extreme samples to keep increasing their influence. The corresponding piecewise-linear behavior is shown in Figure 2.4.

The pinball loss modifies the hinge loss by penalizing both sides of the margin [27]:

$$\mathfrak{L}_{\text{pin}}(u) = \begin{cases} u, & u > 0, \\ -\tau u, & u \leq 0, \end{cases} \qquad 0 \leq \tau \leq 1. \tag{2.10}$$

For $\tau = 0$, the pinball loss reduces to the hinge loss. For $\tau > 0$, correctly classified samples outside the margin receive a flexible penalty controlled by τ, which improves the noise-

Table 2.2: Summary of representative classification loss functions for margin-based learning.

Loss	**Formulation**	**Main properties**
Hinge loss	$\mathfrak{L}(u) = \max(0, u)$	Convex, non-smooth, sparse, unbounded
Pinball loss	$\mathfrak{L}(u) = \begin{cases} u, & u > 0, \\ -\tau u, & u \leq 0 \end{cases}$	Convex, non-smooth, non-sparse for $\tau > 0$, unbounded
Squared hinge loss	$\mathfrak{L}(u) = \max(0, u)^2$	Convex, smooth, sparse, unbounded
Smooth pinball loss	$\mathfrak{L}(u) = \begin{cases} (\tau u)^2, & u > 0, \\ ((1-\tau)u)^2, & u \leq 0 \end{cases}$	Convex, smooth, asymmetric, unbounded
Ramp loss	$\mathfrak{L}(u) = \begin{cases} \theta, & u \geq \theta, \\ u, & 0 < u < \theta, \\ 0, & u \leq 0 \end{cases}$	Bounded, sparse, non-convex, non-smooth
Truncated pinball loss	$\mathfrak{L}(u) = \begin{cases} \delta_1, & u \geq \delta_1, \\ u, & 0 \leq u < \delta_1, \\ -\tau u, & -\delta_2/\tau < u < 0, \\ \delta_2, & u \leq -\delta_2/\tau \end{cases}$	Bounded, asymmetric, non-convex, non-smooth
LINEX loss	$\mathfrak{L}(u) = \exp(au) - au - 1, a \neq 0$	Convex, smooth, asymmetric, unbounded
BLINEX loss	$\mathfrak{L}(u) = \begin{cases} \exp(au) - au - 1, & u < \theta, \\ \exp(a\theta) - a\theta - 1, & u \geq \theta \end{cases}$	Smooth, asymmetric, bounded through saturation

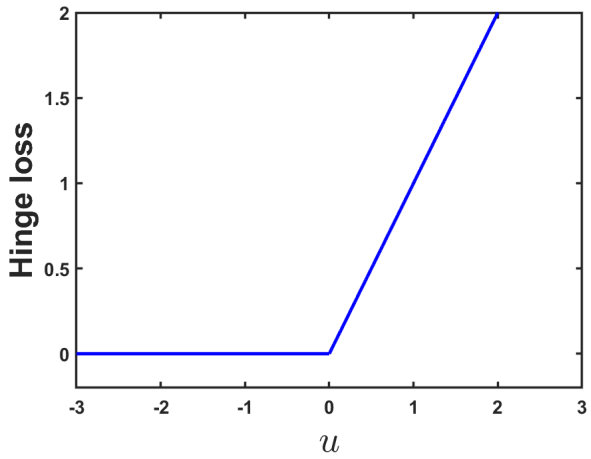


Figure 2.4: Hinge loss. The loss is zero for correctly classified samples outside the margin and grows linearly for margin violations.

insensitive behavior of the loss around the decision boundary. Similar to hinge loss, pinball loss is convex, non-smooth, and unbounded, as illustrated in Figure 2.5.

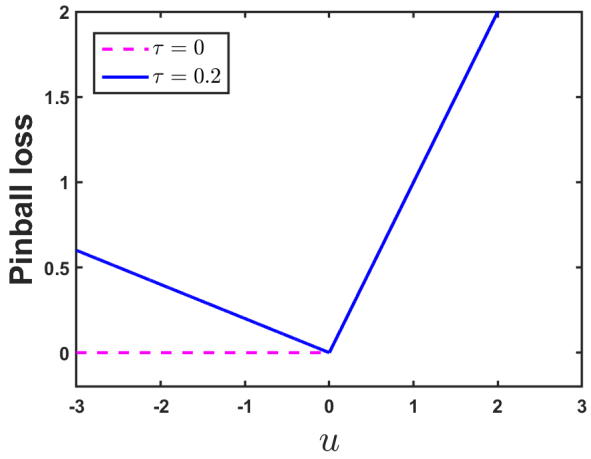


Figure 2.5: Pinball loss. The parameter τ controls the penalty assigned to samples on the correctly classified side of the margin.

2.3.2 Convex and Smooth Losses: Squared Hinge, Smooth Pinball, and LINEX

The squared hinge loss, also used in L_2-loss SVM formulations, is [28]

$$\mathfrak{L}_{\text{s-hinge}}(u) = \begin{cases} u^2, & u > 0, \\ 0, & u \leq 0. \end{cases} \tag{2.11}$$

It is convex and differentiable at the margin transition, so it is smoother than hinge loss and thus allows the use of gradient-based optimization algorithms. However, its quadratic growth makes it more sensitive to outliers than hinge loss. Figure 2.6 shows the resulting quadratic penalty for margin violations.

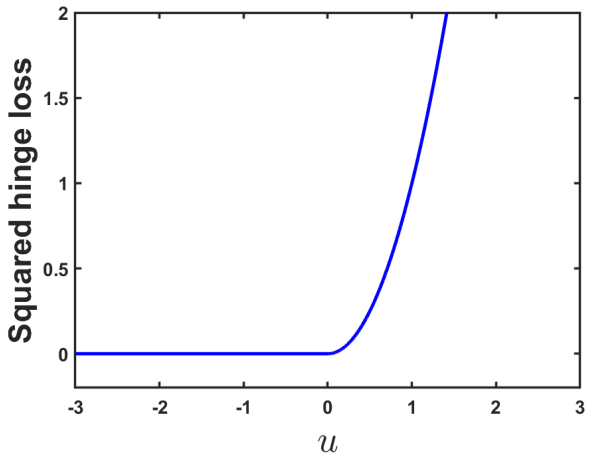


Figure 2.6: Squared hinge loss. The quadratic penalty emphasizes large margin violations, which can improve fitting but increases outlier sensitivity.

The smooth pinball loss has the form [29, 30]

$$\mathfrak{L}_{\text{s-pin}}(u) = \begin{cases} (\tau u)^2, & u > 0, \\ ((1-\tau)u)^2, & u \leq 0, \end{cases} \qquad 0 \leq \tau \leq 1. \tag{2.12}$$

It introduces convex, smooth, and asymmetric quadratic behavior and can be tuned to control the relative treatment of the two sides of the margin. Like squared hinge loss, it is still

unbounded. Figure 2.7 displays this asymmetric smooth shape.

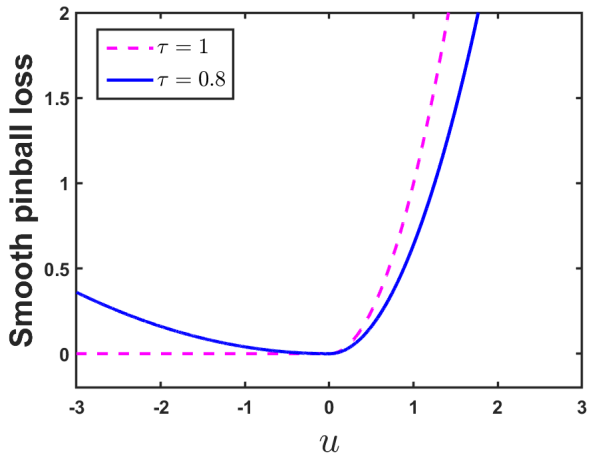


Figure 2.7: Smooth pinball loss. The parameter τ controls asymmetric quadratic penalties on the two sides of the margin.

The LINEX loss is

$$\mathfrak{L}_{\text{LINEX}}(u) = e^{au} - au - 1, \qquad a \neq 0. \tag{2.13}$$

It is convex, smooth, and asymmetric; the sign and magnitude of a determine which side of the error receives stronger penalty [31, 32]. This is valuable for cost-sensitive settings, but the exponential term makes the loss unbounded. Figure 2.8 shows the asymmetric growth controlled by a.

2.3.3 Non-Convex and Non-Smooth Losses: Ramp and Truncated Pinball

The ramp loss truncates the hinge loss [33, 34]:

$$\mathfrak{L}_{\text{ramp}}(u) = \begin{cases} \theta, & u \geq \theta, \\ u, & 0 < u < \theta, \\ 0, & u \leq 0, \end{cases} \qquad \theta \geq 1. \tag{2.14}$$

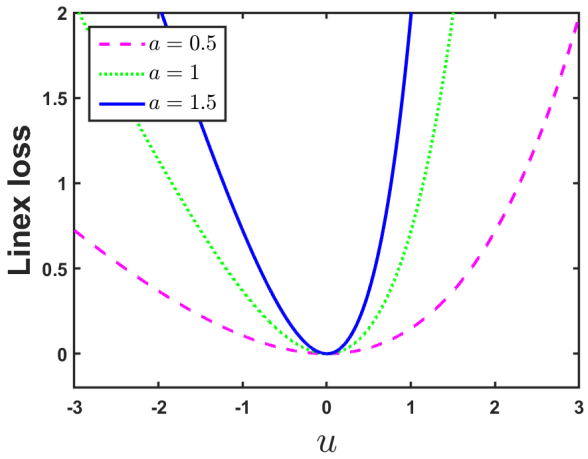


Figure 2.8: LINEX loss. The parameter a controls asymmetric growth, allowing one side of the margin violation to be penalized more heavily.

Its boundedness makes it more robust to outliers than hinge loss. The price is nonconvexity and non-smooth transitions at the truncation points, which complicate optimization. Figure 2.9 shows the flat upper segment created by this truncation.

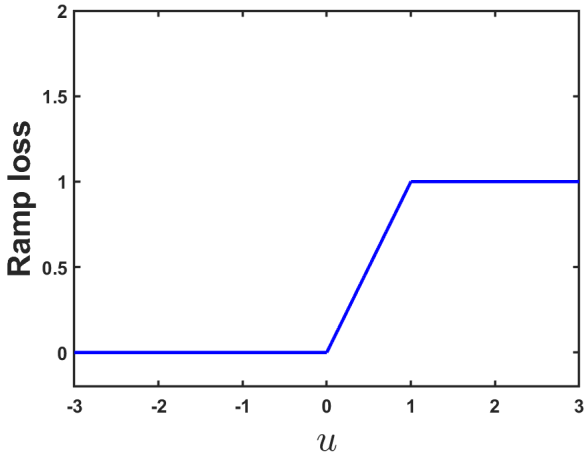


Figure 2.9: Ramp loss. Hard truncation caps the influence of severe margin violations but introduces non-smooth points.

The truncated pinball loss combines the asymmetric behavior of pinball loss with

bounded saturation:

$$\mathfrak{L}_{\text{tpin}}(u) = \begin{cases} \delta_1, & u \geq \delta_1, \\ u, & 0 \leq u < \delta_1, \\ -\tau u, & -\delta_2/\tau < u < 0, \\ \delta_2, & u \leq -\delta_2/\tau, \end{cases} \qquad 0 < \tau \leq 1,\ \delta_1, \delta_2 > 0. \tag{2.15}$$

The upper cap δ_1 limits the influence of severe margin violations, while the lower cap δ_2 prevents correctly classified samples far from the margin from continuing to affect the objective. Hence, truncated pinball loss is bounded and can be more robust and sparse than pinball loss, but it is non-convex and non-smooth at the truncation points [35, 36]. Figure 2.10 illustrates both saturation regions.

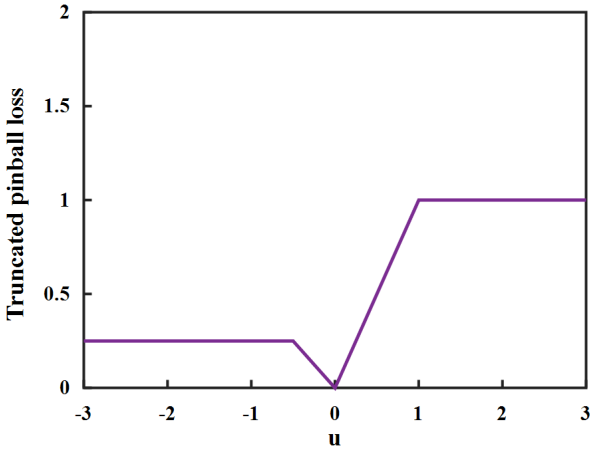


Figure 2.10: Truncated pinball loss with $\tau = 0.5$, $\delta_1 = 1$, and $\delta_2 = 0.25$. The two caps bound the loss on both sides of the margin violation axis.

2.3.4 Deep-Learning Losses for Class Imbalance and Label Noise

The preceding discussion concerns margin losses used primarily in hyperplane-based learning. A parallel literature in deep classification defines the loss through the predicted probability p_y of the true class. Table 2.3 summarizes four representative losses that address

class imbalance or label noise in this probability-based setting.

Table 2.3: Representative probability-based losses for deep classification under class imbalance or label noise.

Loss	**Formulation**	**Main properties**
Focal loss [37]	$\mathcal{L}_{\text{focal}}(p_y) = -(1-p_y)^{\gamma}\log p_y,\ \gamma \geq 0$	Down-weights easy samples; designed for class imbalance; unbounded as $p_y \to 0$
Generalized cross-entropy loss [38]	$\mathcal{L}_q(p_y) = (1-p_y^q)/q,\ 0 < q \leq 1$	Bounded for $q > 0$; interpolates between cross-entropy and mean absolute error
Symmetric cross-entropy loss [39]	$\mathcal{L}_{\text{SCE}} = \alpha\mathcal{L}_{\text{CE}}(\mathbf{y},\mathbf{p}) + \beta\mathcal{L}_{\text{RCE}}(\mathbf{p},\mathbf{y})$	Combines learning efficiency with a noise-tolerant reverse term; requires clipping of one-hot targets
Bi-tempered logistic loss [40]	$-\log_{t_1}(p_y) + \frac{\sum_j p_j^{2-t_1}-1}{2-t_1},\ p_j = \exp_{t_2}(a_j-\nu)$	Bounded matching loss for $t_1 < 1$; heavy-tailed probabilities for $t_2 > 1$; generally non-convex

Focal loss multiplies cross-entropy by $(1-p_y)^{\gamma}$, thereby suppressing the cumulative contribution of abundant easy samples and focusing optimization on difficult examples [37]. It is particularly relevant to class imbalance, but it does not cap the penalty assigned to a confidently incorrect label. Generalized cross-entropy instead uses a Box–Cox-type transformation that approaches cross-entropy as $q \to 0$ and mean absolute error at $q = 1$; for $q > 0$, its bounded range controls the effect of extremely small true-class probabilities [38]. Symmetric cross-entropy augments conventional cross-entropy with reverse cross-entropy, where $\mathcal{L}_{\text{RCE}}(\mathbf{p},\mathbf{y}) = -\sum_j p_j \log y_j$ is evaluated after clipping zero entries of the one-hot target distribution [39]. In the bi-tempered formulation, $\log_t(z) = (z^{1-t}-1)/(1-t)$, $\exp_t$ denotes its inverse on the valid domain, and ν normalizes the class probabilities. The loss

uses tempered logarithmic and exponential functions so that bounded matching and heavy-tailed class probabilities can be controlled separately [40].

These probability-based losses and the margin-based losses developed later address related problems through different interfaces. Focal loss primarily reweights easy and hard probability predictions, while generalized and bi-tempered losses limit sensitivity to noisy labels through bounded or tempered probability penalties. RoBoSS and Wave operate directly on signed-margin violations in SVM- and TSVM-type models: RoBoSS additionally targets a sparse zero-loss region, whereas Wave uses a bounded smooth asymmetric profile to retain controlled influence on both sides of the margin. Thus, the thesis contributions complement rather than duplicate the robust-loss literature developed for deep probability models.

2.3.5 Automated Loss-Function Discovery

Loss functions can also be discovered automatically instead of being specified analytically. Genetic Loss-function Optimization (GLO) represents candidate losses as expression trees, evolves their structure through genetic programming, and tunes coefficients using the covariance matrix adaptation evolution strategy. This procedure discovered the Baikal loss, whose non-monotone shape discourages overconfident predictions and acts as an implicit regularizer [41]. TaylorGLO replaces discrete expression trees with multivariate Taylor-polynomial parameterizations, enabling a smoother and more scalable evolutionary search [42]. Such methods can adapt a loss to a target architecture and validation criterion, but they require an additional outer search, can overfit the meta-objective, and do not automatically provide properties such as boundedness, calibration, sparsity, or an insensitive zone. In contrast, the losses studied in this thesis are constructed from explicit analytical requirements and are accompanied by property or generalization analyses. Automated discovery

and analytical design are therefore complementary routes to robust loss construction.

2.4 Regression Loss Functions

Regression losses are functions of the residual $r = y - f(\mathbf{x})$. The central issue is influence: if $\mathcal{L}(r)$ or its derivative grows without bound, a small number of extreme residuals can dominate the fitted regressor. The following losses appear frequently in robust regression, SVR, LSSVR, and randomized-network learning.

Table 2.4 provides a compact summary of the regression losses used later for comparison and motivation. The detailed discussion that follows explains how each loss trades off robustness, smoothness, boundedness, and insensitivity to small residuals.

2.4.1 Unbounded Regression Losses

The least-squares loss is

$$\mathcal{L}_2(r) = r^2. \tag{2.16}$$

It is smooth, convex, and analytically convenient, which explains its use in LSSVM, LSSVR, RVFL, and BLS. Its weakness is quadratic growth: a residual twice as large receives four times the penalty, making it highly sensitive to outliers. This growth pattern is shown in Figure 2.11.

The absolute loss is

$$\mathcal{L}_1(r) = |r|. \tag{2.17}$$

It grows linearly and is therefore more robust than squared loss for large residuals. However, it is non-smooth at $r = 0$, which can complicate optimization. Figure 2.12 shows the associated V-shaped loss.

Table 2.4: Summary of representative regression loss functions.

Loss	**Formulation**	**Main properties**
ε-insensitive loss	$\mathcal{L}(r) = \max\{0, \lvert r\rvert - \varepsilon\}$	Insensitive zone, non-smooth, unbounded
Least-squares loss	$\mathcal{L}(r) = r^2$	Convex, smooth, unbounded, outlier-sensitive
Absolute loss	$\mathcal{L}(r) = \lvert r\rvert$	Convex, non-smooth, unbounded
Huber loss	$\mathcal{L}(r) = \begin{cases} \frac{1}{2}r^2, & \lvert r\rvert < \theta, \\ \theta\lvert r\rvert - \frac{1}{2}\theta^2, & \lvert r\rvert \geq \theta \end{cases}$	Convex, partially smooth, unbounded
Canal loss	$\mathcal{L}(r) = \min\{\theta - \varepsilon, \max(0, \lvert r\rvert - \varepsilon)\}$	Bounded, insensitive, non-smooth
Ramp-insensitive least-square loss	$\mathcal{L}(r) = \begin{cases} 0, & \lvert r\rvert < \varepsilon, \\ (\lvert r\rvert - \varepsilon)^2, & \varepsilon \leq \lvert r\rvert \leq \theta, \\ (\theta - \varepsilon)^2, & \lvert r\rvert > \theta \end{cases}$	Bounded, insensitive, non-smooth
Bounded least-squares loss	$\mathcal{L}(r) = 1 - \frac{1}{1+\theta r^2}$	Bounded, smooth, no insensitive zone
BLINEX loss	$\mathcal{L}(r) = 1 - \frac{1}{1+\theta(\exp(ar)-ar-1)}, a \neq 0$	Bounded, smooth, asymmetric, no insensitive zone

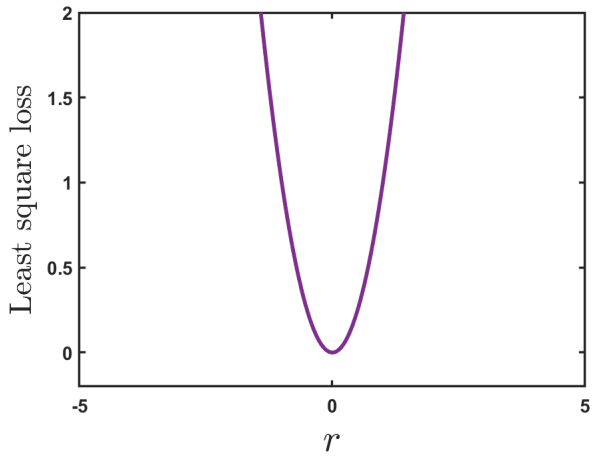


Figure 2.11: Least-squares loss. Quadratic growth makes optimization convenient but gives large residuals excessive influence.

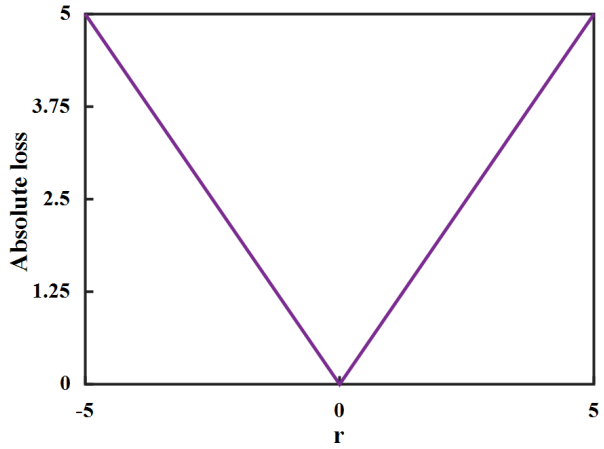


Figure 2.12: Absolute loss. Linear growth improves robustness relative to least squares, but the cusp at zero is non-smooth.

The Huber loss combines squared and absolute behavior [4]:

$$\mathcal{L}_\theta^{\text{Huber}}(r) = \begin{cases} \frac{1}{2}r^2, & |r| < \theta, \\ \theta|r| - \frac{1}{2}\theta^2, & |r| \geq \theta. \end{cases} \tag{2.18}$$

Small residuals are treated quadratically for smooth fitting, while large residuals receive only linear growth. Huber loss is therefore a compromise between efficiency and robustness, but it remains unbounded. Figure 2.13 illustrates this quadratic-to-linear transition.

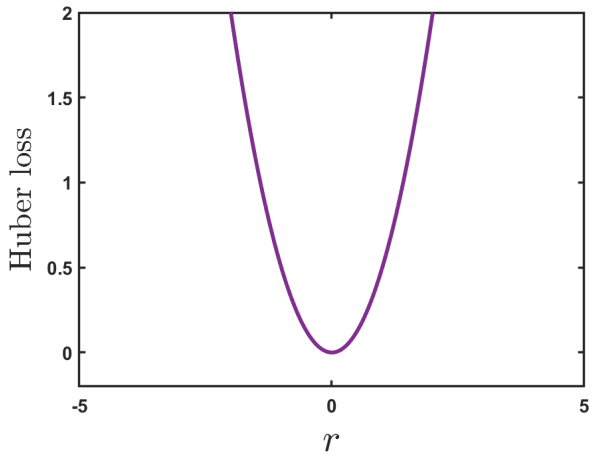


Figure 2.13: Huber loss. The loss is quadratic near zero and linear for large residuals, balancing smoothness and robustness.

The ε-insensitive loss used in SVR is [5]

$$\mathcal{L}_{\varepsilon}(r) = \max(0, |r| - \varepsilon). \tag{2.19}$$

This is the standard SVR loss. Residuals within $[-\varepsilon, \varepsilon]$ are ignored, producing sparse support-vector solutions. Outside the tube, the loss grows linearly and remains unbounded. Figure 2.14 shows the zero-loss tube in residual space.

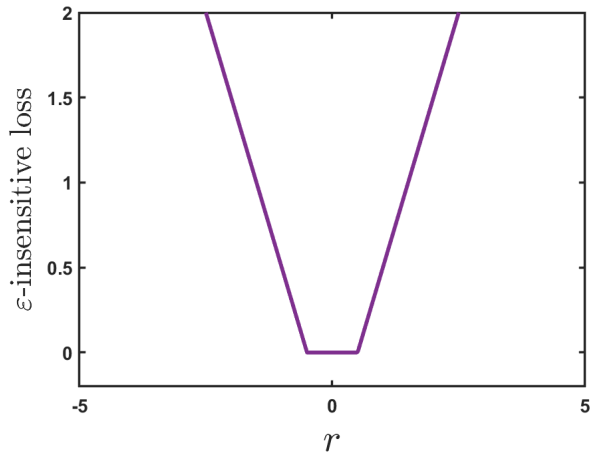


Figure 2.14: ε-insensitive loss. The zero-loss tube improves sparsity and tolerance to small deviations.

2.4.2 Bounded and Insensitive Regression Losses

The ramp-insensitive loss truncates the ε-insensitive loss [43]:

$$\mathcal{L}_{\text{ramp-}\varepsilon}(r) = \begin{cases} 0, & |r| < \varepsilon, \\ |r| - \varepsilon, & \varepsilon \leq |r| \leq \theta, \\ \theta - \varepsilon, & |r| > \theta. \end{cases} \tag{2.20}$$

It preserves the insensitive zone and caps the effect of extreme residuals. Its hard truncation makes it non-smooth at the breakpoints. Figure 2.15 shows the combination of the zero tube

and upper cap.

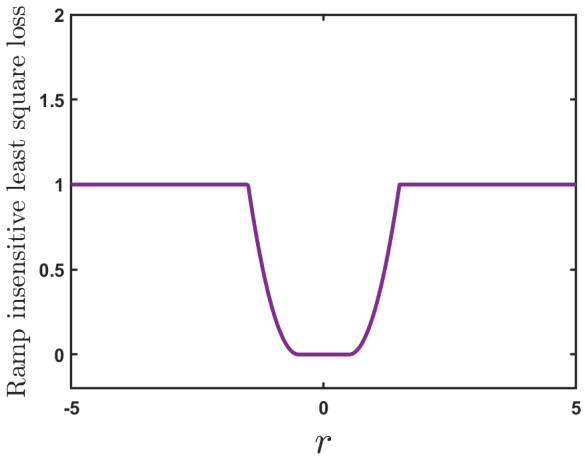


Figure 2.15: Ramp-insensitive least-square-type loss. The insensitive zone and upper cap improve robustness, while hard transitions introduce non-smoothness.

The nonconvex least-squares loss is [44]

$$\mathcal{L}_{\text{ncls}}(r) = \begin{cases} r^2, & |r| \le \theta, \\ \theta^2, & |r| > \theta. \end{cases} \tag{2.21}$$

It behaves like least squares for ordinary residuals but caps the influence of extreme residuals. The cap improves robustness but introduces nonconvexity and non-smoothness at $|r| = \theta$. Figure 2.16 shows the capped quadratic shape.

The quadratic nonconvex insensitive loss uses an insensitive zone, a quadratic middle region, and a bounded/controlled outer region:

$$\mathcal{L}_{\text{qnci}}(r) = \begin{cases} 0, & |r| < \varepsilon, \\ (|r| - \varepsilon)^2, & \varepsilon \le |r| \le t, \\ (t - \varepsilon)^2 + \theta|r| - \theta t, & |r| > t. \end{cases} \tag{2.22}$$

This family emphasizes moderate residuals while reducing sensitivity to small deviations. Depending on parameter choices, it can reduce outlier influence more effectively than pure

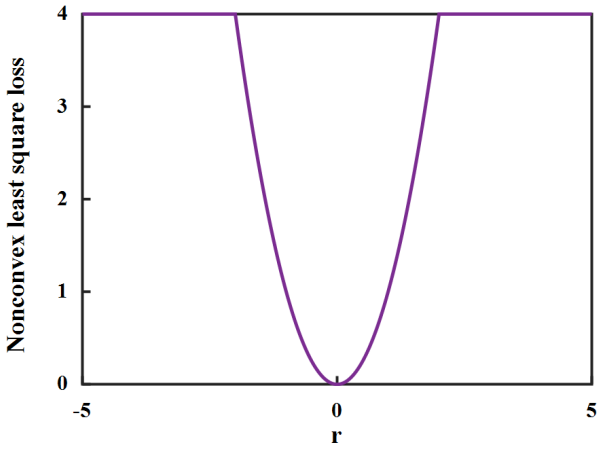


Figure 2.16: Nonconvex least-squares loss. Ordinary residuals are penalized quadratically, while extreme residuals are capped.

squared loss [45]. Figure 2.17 visualizes the zero tube, middle quadratic part, and outer region.

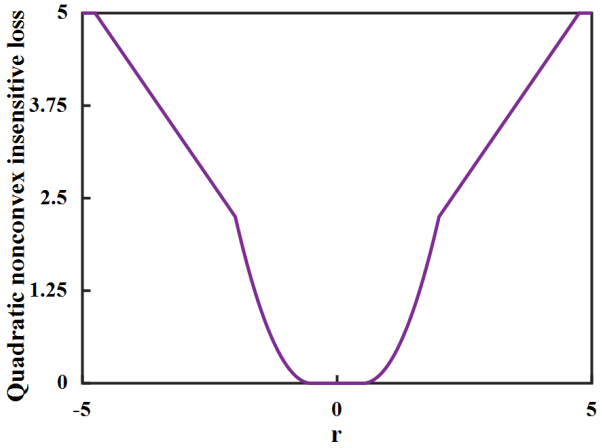


Figure 2.17: Quadratic nonconvex insensitive loss. The loss combines a zero tube, quadratic middle behavior, and controlled outer growth.

The canal loss is

$$\mathcal{L}_{\text{canal}}(r) = \min\{\theta - \varepsilon, \max(0, |r| - \varepsilon)\}. \tag{2.23}$$

It is a capped version of the ε-insensitive loss. It is robust and sparse but non-smooth because

of the max-min structure [46]. Figure 2.18 shows the capped insensitive profile.

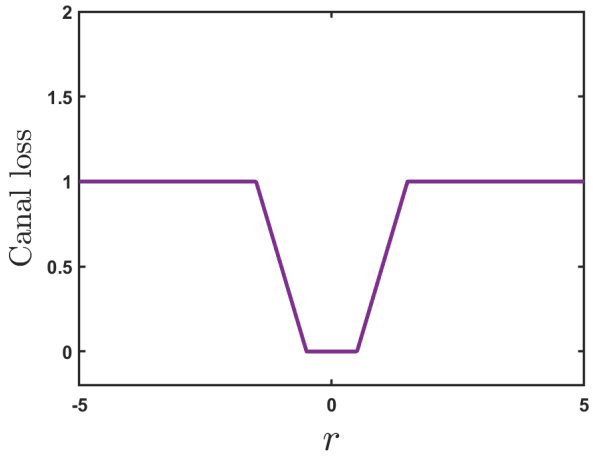


Figure 2.18: Canal loss. The loss keeps the SVR-style zero tube and caps large residuals at a fixed level.

2.4.3 Smooth Bounded Regression Losses

The bounded least-squares loss is [47]

$$\mathcal{L}_{\mathrm{BLSq}}(r) = 1 - \frac{1}{1+\theta r^2}, \qquad \theta > 0. \tag{2.24}$$

It replaces hard truncation with smooth saturation. This bounds the influence of large residuals while maintaining differentiability, but it does not include an insensitive zone. Figure 2.19 shows the gradual saturation.

The BLINEX loss combines asymmetric LINEX behavior with bounded saturation [48, 49]:

$$\mathcal{L}_{\mathrm{BLINEX}}(r) = 1 - \frac{1}{1+\theta(e^{ar} - ar - 1)}, \qquad a \neq 0,\ \theta > 0. \tag{2.25}$$

It is smooth, bounded, and asymmetric, making it suitable when positive and negative residuals have different costs [49]. Like bounded least squares, it lacks an explicit insensitive tube. As shown in Figure 2.20, the sign and magnitude of a control the direction and strength

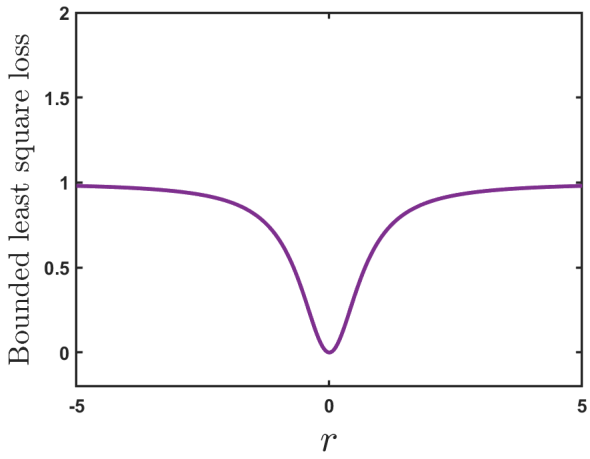


Figure 2.19: Bounded least-squares loss. Smooth saturation limits the effect of large residuals without using hard truncation.

of asymmetry, while θ controls how quickly the curve approaches its upper bound.

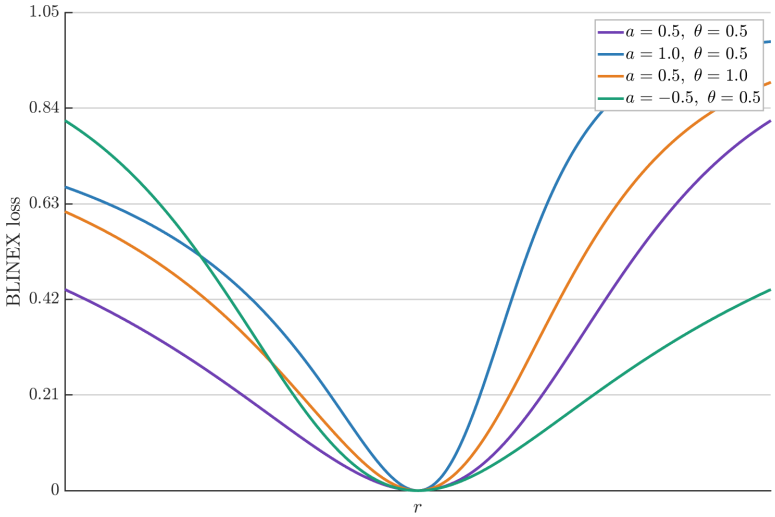


Figure 2.20: BLINEX loss for different values of a and θ. The parameter a controls asymmetry, while θ changes the saturation behavior.

2.4.4 Broader Robust-Regression Perspective

Robust regression extends beyond SVR-specific losses to redescending M-estimators, whose derivatives decrease toward zero for sufficiently extreme residuals [50]. Representative examples are summarized in Table 2.5. Tukey's biweight is constant outside a cutoff

and therefore completely rejects sufficiently large residuals. The Welsch loss approaches its upper bound smoothly, while the adaptive family of Barron [51] continuously recovers several classical penalties by changing a shape parameter. These losses provide bounded or redescending influence, but none contains an exact interval of zero loss around the origin. Consequently, robust M-estimation does not remove the specific design gap considered in Chapter 5: simultaneous smoothness, boundedness, and an explicit insensitive zone.

Table 2.5: Representative robust regression losses beyond the immediate SVR literature.

Loss	**Formulation**	**Main properties**
Tukey biweight	$\mathcal{L}(r) = \begin{cases} \frac{c^2}{6}\left[1-\left(1-(r/c)^2\right)^3\right], & \lvert r\rvert \le c, \\ \frac{c^2}{6}, & \lvert r\rvert > c, \end{cases}$	Bounded, smooth at the cutoff, redescending, no insensitive zone
Welsch loss	$\mathcal{L}(r) = \frac{c^2}{2}\left[1-\exp\left(-(r/c)^2\right)\right]$	Bounded, smooth, redescending, no insensitive zone
Adaptive robust loss [51]	$\mathcal{L}(r;\alpha,c) = \frac{\lvert\alpha-2\rvert}{\alpha}\left[\left(1+\frac{(r/c)^2}{\lvert\alpha-2\rvert}\right)^{\alpha/2}-1\right]$	Smooth shape family; includes quadratic, Charbonnier, Cauchy, and Welsch limits

The preceding classification and regression losses reveal several limitations that motivate further loss-function development. In classification, losses that are both sparse and bounded, such as ramp-type and truncated losses, are usually non-smooth and non-convex; hence, the resulting optimization problem becomes difficult to solve with standard gradient-based procedures. Existing sparse bounded losses also commonly contain non-differentiable transition points, including the origin or truncation boundaries. There is therefore a need for classification losses that can jointly provide sparsity, bounded influence, and smoothness. Noise-insensitive losses such as pinball loss provide flexible treatment of samples around the margin, but they remain non-smooth and unbounded. Truncated pinball loss

adds boundedness, but the hard truncation makes the loss non-convex and non-smooth, which complicates optimization. Hence, robust classification under label noise and outliers requires losses that combine noise insensitivity, boundedness, and smoothness so that efficient gradient-based optimization can be used. In regression, the key difficulty is that losses with an insensitive zone are typically non-smooth at the boundary of the insensitive region, whereas smooth bounded losses generally do not include an insensitive zone. This leaves a gap for regression losses that are simultaneously smooth, bounded, and insensitive to small residuals.

2.5 Class Imbalance, Noise, and Sample Weighting

Real-world classification problems often exhibit class imbalance, label noise, outliers, and local ambiguity. Class imbalance occurs when one class has substantially fewer samples than another. A classifier trained without correcting for imbalance may achieve high overall accuracy while failing to recognize minority-class samples. Noise and outliers further degrade performance by distorting margins, class centers, and learned decision boundaries.

2.5.1 Data-Level and Algorithm-Level Strategies

Imbalance learning methods are broadly divided into data-level and algorithm-level strategies. Data-level methods modify the training set through under-sampling, over-sampling, or synthetic sample generation. SMOTE is a representative over-sampling method that generates synthetic minority samples by interpolation between minority samples and their nearest minority neighbors [11]. Algorithm-level methods modify the learning objective through class-dependent costs, fuzzy memberships, local weighting, or ensemble construction.

Cost-sensitive learning increases the penalty for misclassifying minority-class samples.

Fuzzy SVM assigns a membership value μ_i to each sample so that less reliable samples contribute less to the classifier [12]. A weighted empirical risk minimization problem has the form

$$\min_{\theta} \sum_{i=1}^{m} \mu_i \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big) + \lambda \Omega(\theta), \tag{2.26}$$

where $\mu_i \in [0, \infty)$ measures the reliability or importance of the ith sample. In weighted LSSVM, for example, sample-dependent penalties can be incorporated as

$$\begin{aligned} \min_{\mathbf{w},b,\mathbf{e}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{m} \mu_i e_i^2, \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) = 1 - e_i, \quad i = 1, \ldots, m. \end{aligned} \tag{2.27}$$

Large μ_i increases the influence of a sample, while small μ_i reduces its effect on the learned decision function.

2.5.2 Classical Fuzzy Membership Schemes

Distance-to-center schemes assign higher membership to samples near the class center and lower membership to distant samples. Let $\mathcal{C}_{y_i}$ be the center of the class to which $\mathbf{x}_i$ belongs, $d_i^{\text{cen}} = \|\mathbf{x}_i - \mathcal{C}_{y_i}\|_2$, and $R_{y_i} = \max_{\ell : y_\ell = y_i} \|\mathbf{x}_\ell - \mathcal{C}_{y_i}\|_2$. Representative center-based memberships are

$$\mu_{\text{lin}}^{\text{cen}}(\mathbf{x}_i) = 1 - \frac{d_i^{\text{cen}}}{R_{y_i} + \delta}, \tag{2.28}$$

$$\mu_{\text{exp}}^{\text{cen}}(\mathbf{x}_i) = \frac{2}{1 + \exp(\gamma d_i^{\text{cen}})}, \qquad \gamma \in [0, 1], \tag{2.29}$$

where $\delta > 0$ avoids zero membership. Hyperplane-based schemes first construct a preliminary classifier $f_0(\mathbf{x})$ and then assign membership using the distance to that hyperplane:

$$\mu_{\text{lin}}^{\text{hyp}}(\mathbf{x}_i) = 1 - \frac{|f_0(\mathbf{x}_i)|}{\max_\ell |f_0(\mathbf{x}_\ell)| + \delta}, \tag{2.30}$$

$$\mu_{\text{exp}}^{\text{hyp}}(\mathbf{x}_i) = \frac{2}{1 + \exp(\gamma |f_0(\mathbf{x}_i)|)}. \tag{2.31}$$

Later fuzzy and intuitionistic fuzzy methods incorporate information such as class boundary proximity, entropy, local density, or class affinity [13, 52, 53, 54]. For example, affinity and class-probability based fuzzy SVM first estimates the position of a majority-class sample relative to an SVDD hypersphere:

$$\varphi_{\text{aff}}(\mathbf{x}_i) = \begin{cases} 0.8 \left(\dfrac{d_i^{\text{svdd}} - \min(d_i^{\text{svdd}})}{\max(d_i^{\text{svdd}}) - \min(d_i^{\text{svdd}})} \right), & d_i^{\text{svdd}} < \beta R, \\ 0.2 \exp\left(\rho \left(1 - \dfrac{d_i^{\text{svdd}}}{\beta R} \right) \right), & d_i^{\text{svdd}} \geq \beta R, \end{cases} \tag{2.32}$$

where R is the SVDD radius, d_i^{svdd} is the distance from the SVDD center, $\beta \in (0, 1]$, and $\rho > 0$. A local class-probability term is then computed from the k-nearest neighborhood,

$$p(\mathbf{x}_i) = \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)} \mathbf{1}\{y_j = y_i\}, \tag{2.33}$$

and the membership can be written as

$$\mu_{\text{aff}}(\mathbf{x}_i) = \begin{cases} 1, & \mathbf{x}_i \in \text{minority class}, \\ \varphi_{\text{aff}}(\mathbf{x}_i) p(\mathbf{x}_i), & \mathbf{x}_i \in \text{majority class}. \end{cases} \tag{2.34}$$

This type of design explicitly combines geometric reliability with local label agreement, but it may become computationally expensive because SVDD itself requires solving a quadratic programming problem.

2.5.3 Limitations of Distance-Based Weighting

The effectiveness of a weighting scheme depends on whether its reliability score matches the actual role of a sample in the learning problem. Figure 2.21 illustrates three common difficulties. First, many valid samples can lie near the class boundary, so a center-distance rule may assign them very small weights even though they are important for defining the decision surface; this case is shown in Figure 2.21a. Second, when many reliable samples form a dense neighborhood around the class center, an aggressively decaying function may still distinguish among samples that should receive nearly equal confidence, as shown in Figure 2.21b. Third, a noisy or mislabeled sample can have the same distance from the class center as a clean sample, so distance alone cannot identify label inconsistency; this ambiguity is shown in Figure 2.21c.

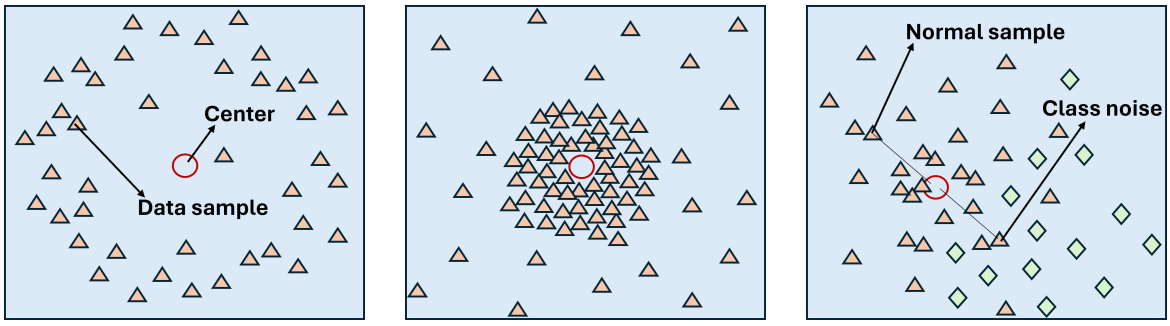


Figure 2.21: Challenges for fuzzy weighting in real-world data. (a) Many valid samples can be far from the class center. (b) Central samples may deserve similar reliability despite small distance differences. (c) Noisy samples can share the same center distance as clean samples, requiring information beyond distance alone.

These limitations explain why modern weighting schemes often combine several signals: global geometry, local label consistency, class priors, boundary proximity, and sometimes density or entropy. A purely distance-based rule is simple and efficient, but it may be unreliable under class overlap, asymmetric class shapes, skewed priors, and label noise. Conversely, richer local or affinity-based rules can improve reliability estimation but may

introduce additional hyperparameters and computational overhead.

2.6 Randomized Neural Networks

Randomized neural networks (RdNNs) reduce training cost by randomly generating part of the network and learning only the output layer. Instead of updating all hidden-layer parameters through backpropagation, these methods sample input-to-hidden weights once and then solve a linear output problem. This makes them attractive for large datasets and repeated experimental evaluation.

2.6.1 Random Vector Functional Link Networks and Extreme Learning Machines

Random vector functional link (RVFL) networks were introduced as randomized single-hidden-layer models with direct links from input to output [16]. Extreme learning machines (ELMs) are closely related randomized single-hidden-layer networks without direct links [17]. Given $X \in \mathbb{R}^{m\times d}$, random weights $W \in \mathbb{R}^{d\times h}$, random bias matrix $B \in \mathbb{R}^{m\times h}$, and activation function $\phi(\cdot)$, the hidden-layer output is

$$H = \phi(XW + B). \tag{2.35}$$

For ELM, the trainable representation is $A = H$. For RVFL, direct input-to-output connections are retained:

$$A = [X \mid H]. \tag{2.36}$$

Figure 2.22 shows this structure: the random hidden representation and the original input are concatenated before output-weight learning. The output weight matrix is learned through

ridge regression:

$$\min_{\Theta} \ \|A\Theta - Y\|_F^2 + \lambda\|\Theta\|_F^2. \tag{2.37}$$

Setting the derivative with respect to Θ to zero gives $2A^\top(A\Theta - Y) + 2\lambda\Theta = 0$, and therefore the primal closed-form solution is

$$\Theta = (A^\top A + \lambda I)^{-1}A^\top Y. \tag{2.38}$$

This form inverts a matrix whose size is determined by the number of representation nodes. When m is smaller than the number of representation nodes, the dual form can be computationally preferable:

$$\Theta = A^\top(AA^\top + \lambda I)^{-1}Y. \tag{2.39}$$

The primal and dual expressions are algebraically equivalent under ridge regularization, but they have different numerical costs. The primal form is usually preferred when the representation dimension is smaller than the number of samples, while the dual form is preferred when the sample size is smaller.

RVFL models are simple and fast, but they are sensitive to random initialization, noise, outliers, and ill-conditioned hidden representations. Robust variants therefore modify the loss, introduce fuzzy weights, use ensembles, regularize the representation, or avoid unstable matrix inversion.

2.6.2 Broad Learning System

The broad learning system (BLS) is a flat randomized architecture designed to avoid deep stacking while retaining representation power [19]. BLS constructs feature mapping nodes and enhancement nodes, concatenates them into a broad feature matrix, and learns the output weights through regularized least squares.

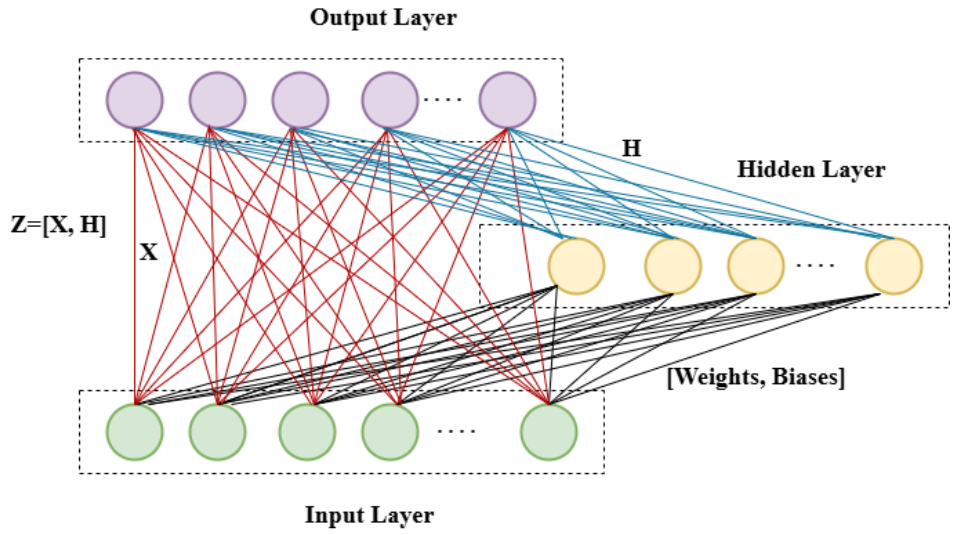


Figure 2.22: RVFL architecture. Hidden-layer parameters are randomly sampled and fixed, while only the output weights are trained. Direct input-to-output links distinguish RVFL from ELM.

Let the feature layer contain q windows, each with p feature nodes. The ith feature window is

$$Z_i = \phi(XW_{f_i} + B_{f_i}), \qquad i = 1, \ldots, q, \tag{2.40}$$

where $W_{f_i} \in \mathbb{R}^{d \times p}$ and $B_{f_i} \in \mathbb{R}^{m \times p}$. Concatenating all feature windows gives

$$Z = [Z_1, Z_2, \ldots, Z_q] \in \mathbb{R}^{m \times pq}. \tag{2.41}$$

The enhancement layer contains s windows, each with r enhancement nodes:

$$E_j = \psi(ZW_{e_j} + B_{e_j}), \qquad j = 1, \ldots, s, \tag{2.42}$$

where $W_{e_j} \in \mathbb{R}^{pq \times r}$. The enhancement output is

$$E = [E_1, E_2, \ldots, E_s] \in \mathbb{R}^{m \times rs}. \tag{2.43}$$

The final broad representation is

$$A = [Z \mid E] \in \mathbb{R}^{m\times(pq+rs)}, \tag{2.44}$$

and the primal output-weight solution is

$$\Theta = (A^\top A + \lambda I)^{-1} A^\top Y. \tag{2.45}$$

The corresponding dual solution is

$$\Theta = A^\top (AA^\top + \lambda I)^{-1} Y. \tag{2.46}$$

The primal form is convenient when the broad representation dimension $pq + rs$ is modest, whereas the dual form can reduce computational cost when $m < pq + rs$. Both forms rely on the conditioning of the matrix being inverted, which is why randomized neural models can become unstable when the generated representation is ill-conditioned or heavily corrupted by noise. The flat feature-window and enhancement-window layout is illustrated in Figure 2.23.

The flat structure of BLS can be efficient and incrementally expandable, but its least-squares training objective can be sensitive to noise and outliers. This has motivated robust BLS variants based on regularization, fuzzy systems, collaborative learning, and alternative objective functions.

2.6.3 Deep RVFL

Deep RVFL extends the RVFL idea by stacking multiple randomized hidden layers while retaining direct links and closed-form output training [20]. With $H^{(0)} = X$, the

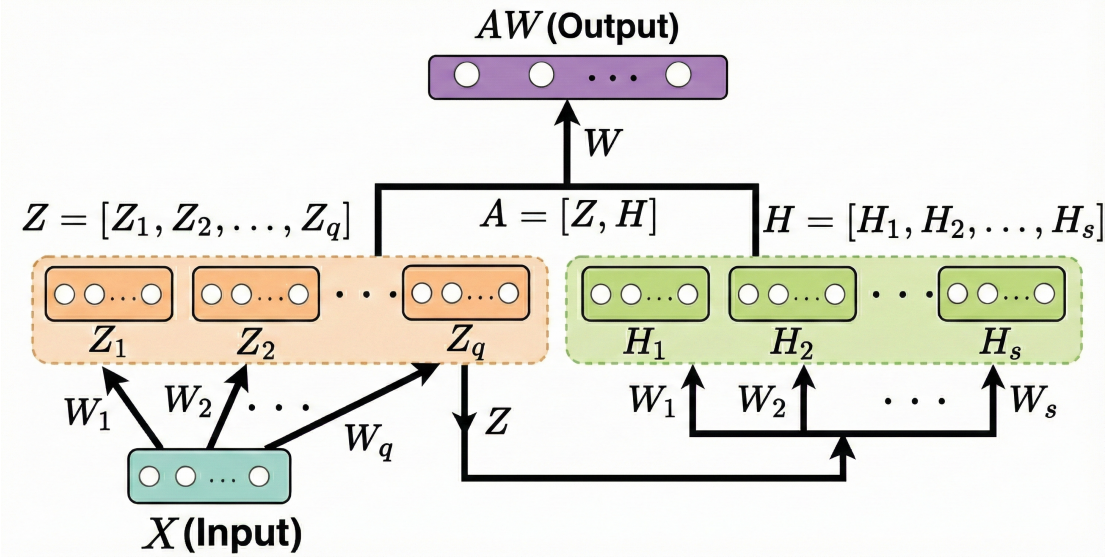


Figure 2.23: BLS architecture. Feature mapping nodes and enhancement nodes are generated in a broad flat structure, then concatenated to train the output weights.

l^{th} random hidden layer is

$$H^{(l)} = \phi\big(H^{(l-1)}W^{(l)} + B^{(l)}\big), \qquad l = 1, \ldots, L. \tag{2.47}$$

An augmented deep representation can be formed as

$$A = [X \mid H^{(1)} \mid H^{(2)} \mid \cdots \mid H^{(L)}], \tag{2.48}$$

followed by the same ridge-type output learning used in (2.38). Figure 2.24 shows the layered structure. Each hidden layer is randomly generated and frozen, while the final readout uses a concatenation of the input and hidden representations. Such architectures improve representation capacity but may also increase sensitivity to random feature quality and numerical conditioning. Recent robust RVFL developments integrate fuzzy inference, granular-ball representations, ensemble strategies, and robust losses [55, 56, 57]. These works show that randomized neural networks can remain computationally efficient while becoming more robust to uncertainty and corrupted data.

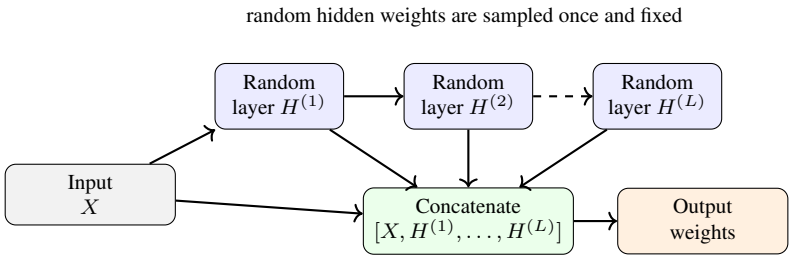


Figure 2.24: Deep RVFL architecture. Multiple random hidden layers are stacked, and the output layer is trained using an augmented representation containing the input and hidden-layer outputs.

Although RVFL, BLS, and Deep RVFL provide efficient alternatives to fully trained neural networks, their standard formulations have three important limitations. First, the output layer is commonly trained using squared-error loss, which can be strongly affected by noisy labels, outliers, and large residuals. Second, closed-form output learning requires matrix inversion or pseudo-inversion; when the hidden or broad representation matrix is ill-conditioned, small perturbations in the data can be amplified in the learned output weights. Third, the random hidden weights are usually generated independently from simple distributions, so the hidden matrix is not explicitly adapted to the dependence structure of the input data. This can reduce representation quality and introduce instability across different random initializations, especially in high-dimensional or correlated datasets.

2.7 Copula-Based Dependency Modeling

Randomized neural networks usually sample hidden weights independently from simple distributions. This independence assumption ignores statistical dependence among input features. If the input features are correlated, tail-dependent, or exhibit nonlinear association, an independent random projection may fail to capture useful joint structure.

Copula theory provides a principled way to separate marginal distributions from dependence structure. A d-variate copula is a distribution function on $[0,1]^d$ whose univariate

marginals are uniform. Thus, instead of modeling feature scales and feature dependence in a single object, copula modeling first maps each feature to its probability scale and then models how these probability-scale variables move together.

By Sklar's theorem, any multivariate distribution with joint cumulative distribution function F and marginals $F_1, \ldots, F_d$ can be represented as

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)), \tag{2.49}$$

where C is a copula [21, 22, 58]. If the marginals are continuous, the copula is unique. Conversely, any copula combined with valid marginals defines a valid joint distribution. Figure 2.25 illustrates this decomposition in the bivariate case.

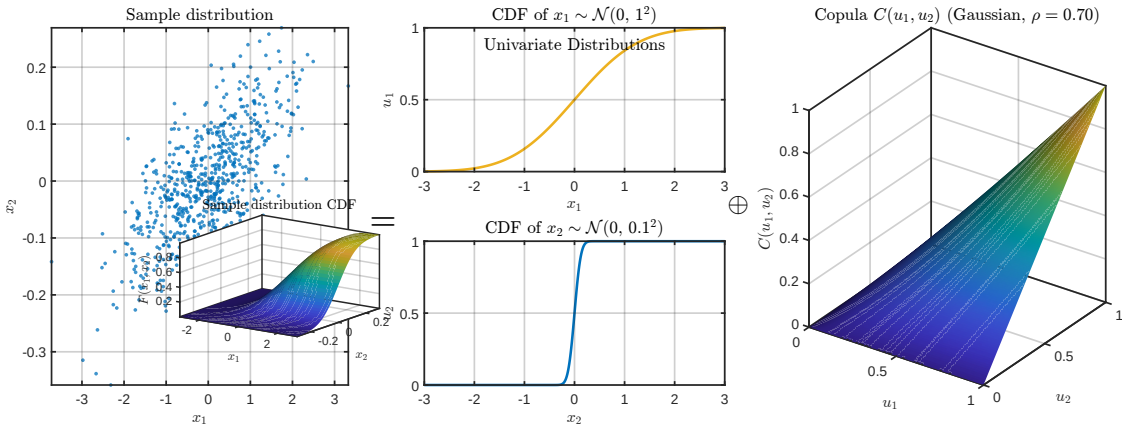


Figure 2.25: Copula decomposition. The empirical variables are transformed through their marginal CDFs to uniform variables, and the copula models the remaining dependence structure on the unit square.

An important property of copulas is invariance under strictly increasing marginal transformations. If each ψ_j is strictly increasing, then $(X_1, \ldots, X_d)$ and $(\psi_1(X_1), \ldots, \psi_d(X_d))$ have the same copula. This makes copulas useful for heterogeneous tabular data, where features may have different physical units, ranges, and marginal distributions. Independence

corresponds to the product copula,

$$C_{\Pi}(u_1, \ldots, u_d) = \prod_{j=1}^{d} u_j, \tag{2.50}$$

whereas non-product copulas encode dependence beyond marginal behavior.

For data X, pseudo-observations can be constructed from feature-wise ranks:

$$u_{ij} = \frac{\mathrm{rank}(x_{ij})}{m+1}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, d. \tag{2.51}$$

The resulting matrix $U = [u_{ij}]$ lies in $(0,1)^{m \times d}$ and captures empirical dependence independent of marginal scale. Copula families differ in the kind of dependence they represent. Gaussian copulas capture symmetric dependence governed by a correlation matrix, while t-copulas additionally model tail co-movement through heavier tails. Clayton copulas emphasize lower-tail dependence, Gumbel copulas emphasize upper-tail dependence, and Frank copulas represent more symmetric Archimedean dependence without strong tail asymmetry. These distinctions matter when feature interactions are nonlinear, asymmetric, or concentrated in extremes.

2.8 Optimization and Numerical Stability

Optimization behavior is central to robust learning. Convex formulations such as standard SVM have strong theoretical guarantees but can be computationally expensive. Least-squares formulations reduce optimization cost but often require solving linear systems, which can become unstable when the associated matrix is ill-conditioned. Robust nonconvex bounded losses improve resistance to outliers but require careful optimization.

Gradient-based optimization methods such as stochastic gradient descent, adaptive gradient methods, and momentum-based methods are widely used for large-scale machine

learning [23, 59, 60]. Nesterov accelerated gradient introduces a look-ahead momentum mechanism that can improve convergence behavior [61]. In robust learning, iterative optimization is often useful when the loss is nonquadratic, nonconvex, or when direct matrix inversion is numerically unstable.

2.9 Evaluation Metrics

This section retains only the definitions needed to interpret the experimental chapters and to fix a common notation; the metrics themselves are standard evaluation tools rather than contributions of the thesis.

Since the thesis studies classification, regression, class imbalance, noisy learning, and biomedical diagnosis, no single metric is sufficient. This section summarizes the metrics used across the empirical chapters.

2.9.1 Classification Metrics

For binary classification, let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives. Accuracy is $\mathrm{Accuracy} = (\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN})$. Sensitivity or recall is $\mathrm{Sensitivity} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$, and specificity is $\mathrm{Specificity} = \mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$. Precision and F1-score are given by $\mathrm{Precision} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ and $\mathrm{F1} = 2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}/(\mathrm{Precision} + \mathrm{Recall})$, respectively. For imbalanced classification, F1-score, sensitivity, specificity, and geometric mean are often more informative than accuracy. The geometric mean is $\mathrm{Gmean} = \sqrt{\mathrm{Sensitivity} \cdot \mathrm{Specificity}}$. Area under the ROC curve (AUC) is also useful when classifier scores are available, because it evaluates ranking quality across thresholds.

For multiclass classification, metrics are computed using macro, micro, or weighted averaging. Macro averaging treats all classes equally, whereas weighted averaging accounts

for class frequency. In imbalanced settings, macro-F1 and class-wise sensitivity are particularly important because they do not allow majority classes to dominate the evaluation.

2.9.2 Regression Metrics

For regression, let y_i be the true response and $\hat{y}_i$ the predicted response. The mean absolute error (MAE) is $\mathrm{MAE} = m^{-1}\sum_{i=1}^{m}|y_i - \hat{y}_i|$, and the root mean squared error (RMSE) is $\mathrm{RMSE} = \sqrt{m^{-1}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$. The coefficient of determination is $R^2 = 1 - \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{m}(y_i - \bar{y})^2$, where $\bar{y}$ is the mean of the observed responses. For robust regression, MAE and related absolute-error measures are often more stable under outliers than squared-error-based measures, while RMSE remains useful for measuring large deviations. For regression experiments where overestimation and underestimation behavior must be inspected separately, positive and negative error measures are also useful. Using $f(\mathbf{x}_i) = \hat{y}_i$, these quantities can be written as

$$\mathrm{Error}_{\mathrm{pos}} = \frac{1}{m}\sum_{i:\, y_i \geq \hat{y}_i} |y_i - \hat{y}_i|, \qquad \mathrm{Error}_{\mathrm{neg}} = \frac{1}{m}\sum_{i:\, y_i < \hat{y}_i} |y_i - \hat{y}_i|.$$

Here, $\mathrm{Error}_{\mathrm{pos}}$ summarizes the magnitude of underprediction errors and $\mathrm{Error}_{\mathrm{neg}}$ summarizes the magnitude of overprediction errors.

2.10 Statistical Comparison of Learning Algorithms

The following tests are stated compactly to establish the exact ranking, significance, and tie conventions used repeatedly in later chapters. Their general statistical foundations follow the cited original and tutorial sources.

When multiple algorithms are evaluated over several datasets, average accuracy alone may be misleading. Statistical tests are therefore used to assess whether observed differ-

ences are likely to be systematic rather than accidental. In this section, let D denote the number of datasets and p denote the number of algorithms or models being compared. This notation is local to the statistical-comparison discussion. In this thesis, statistical analysis is used as an empirical reliability check: the goal is not only to report the best mean performance, but also to verify whether the observed differences persist across datasets or repeated runs.

2.10.1 Friedman and Iman–Davenport Tests

The Friedman test is a nonparametric test for comparing p algorithms over D datasets [24, 25]. For each dataset, the algorithms are ranked separately: the best-performing algorithm receives rank 1, the second receives rank 2, and so on. Ties are assigned average ranks. Let $R_e^{(i)}$ denote the rank of algorithm e on dataset i, and let

$$\bar{R}_e = \frac{1}{D} \sum_{i=1}^{D} R_e^{(i)} \tag{2.52}$$

be its average rank. The null hypothesis states that all algorithms have equivalent performance, meaning their expected ranks are equal. The Friedman statistic is

$$\chi_F^2 = \frac{12D}{p(p+1)} \left[\sum_{e=1}^{p} \bar{R}_e^2 - \frac{p(p+1)^2}{4} \right]. \tag{2.53}$$

Under the null hypothesis and for sufficiently large D and p, χ_F^2 is approximately chi-square distributed with $p-1$ degrees of freedom.

Because the chi-square approximation can be conservative, the Iman–Davenport correction transforms the Friedman statistic into an F-distributed statistic [62]:

$$F_F = \frac{(D-1)\chi_F^2}{D(p-1) - \chi_F^2}. \tag{2.54}$$

This statistic follows an F-distribution with $p-1$ and $(p-1)(D-1)$ degrees of freedom. If F_F exceeds the corresponding critical value at significance level α, the null hypothesis is rejected. The result indicates that at least one algorithm differs significantly from the others, but it does not identify which pairs are different.

2.10.2 Nemenyi Post-Hoc Test

After rejecting the global null hypothesis, the Nemenyi post-hoc test compares all pairs of algorithms using their average ranks [25, 63]. The critical difference (CD) is

$$\mathrm{CD} = q_\alpha \sqrt{\frac{p(p+1)}{6D}}, \tag{2.55}$$

where q_α is the critical value based on the studentized range statistic divided by $\sqrt{2}$, adjusted for the number of algorithms. Two algorithms a and b are considered significantly different if

$$|\bar{R}_a - \bar{R}_b| > \mathrm{CD}. \tag{2.56}$$

Since rank 1 denotes the best performance, the algorithm with the lower average rank is statistically better when the rank difference exceeds the critical difference. The Nemenyi test is especially useful for presenting critical-difference diagrams, where algorithms are placed on an average-rank axis and groups with no significant difference are connected.

2.10.3 Wilcoxon Signed-Rank Test

When comparing two algorithms across matched datasets or repeated folds, the Wilcoxon signed-rank test provides a nonparametric paired comparison [25, 64]. Let $d_i = s_i^{(A)} - s_i^{(B)}$ be the performance difference between algorithms A and B on dataset or run i, where the score s_i is oriented so that a larger value is better. For error measures

such as RMSE or MAE, the sign is reversed before forming d_i. Zero differences are removed. The remaining absolute differences $|d_i|$ are ranked, and each rank is assigned the sign of d_i. The positive and negative rank sums are

$$W^+ = \sum_{d_i>0} \operatorname{rank}(|d_i|), \qquad W^- = \sum_{d_i<0} \operatorname{rank}(|d_i|). \tag{2.57}$$

The test statistic is usually $W = \min(W^+, W^-)$. A small value of W indicates that the differences are consistently in one direction. Unlike a paired t-test, the Wilcoxon signed-rank test does not assume normally distributed performance differences, which makes it appropriate for machine-learning comparisons over heterogeneous datasets.

2.10.4 Win–Tie–Loss Test

The win–tie–loss (WTL) test is a simple pairwise comparison between two algorithms. It is often reported descriptively, and it can also be interpreted through the sign-test approximation when only the direction of pairwise differences is used [25]. For two algorithms A and B, define

$$\operatorname{sgn}_\epsilon(d_i) = \begin{cases} 1, & d_i > \epsilon, \\ 0, & |d_i| \le \epsilon, \\ -1, & d_i < -\epsilon, \end{cases} \tag{2.58}$$

where $d_i = s_i^{(A)} - s_i^{(B)}$, the score is oriented so that larger values are better, and $\epsilon \ge 0$ is a tolerance used to avoid treating numerically negligible differences as wins or losses. The

counts are

$$\begin{aligned} \text{Win} &= \sum_{i=1}^{D} \mathbf{1}\{\text{sgn}_\epsilon(d_i) = 1\}, \\ \text{Tie} &= \sum_{i=1}^{D} \mathbf{1}\{\text{sgn}_\epsilon(d_i) = 0\}, \\ \text{Loss} &= \sum_{i=1}^{D} \mathbf{1}\{\text{sgn}_\epsilon(d_i) = -1\}. \end{aligned} \tag{2.59}$$

If ties are present, the tied datasets are shared equally between the two algorithms. Thus, the effective number of wins for algorithm A is

$$\text{Win}_A^{\text{eff}} = \text{Win} + \frac{1}{2}\text{Tie}. \tag{2.60}$$

Under the null hypothesis that the two algorithms are equivalent, each algorithm is expected to win on $D/2$ datasets. Using the normal approximation to the binomial sign test at the 5% level, algorithm A is considered significantly better than algorithm B if

$$\text{Win}_A^{\text{eff}} \geq \frac{D}{2} + 1.96\frac{\sqrt{D}}{2}. \tag{2.61}$$

The analogous lower-tail condition applies when algorithm B has the effective win count exceeding the same threshold. WTL is useful alongside Friedman, Nemenyi, and Wilcoxon tests because it shows how often a method improves over a baseline across datasets; however, it does not use the magnitude of the differences, so it should be interpreted together with rank-based or signed-rank tests.

2.11 Positioning of the Thesis

The literature reviewed in this chapter reveals related but distinct gaps across loss design, sample weighting, numerical optimization, and randomized representation learning.

Table 2.6 positions the thesis against the closest methodological families and states the unresolved issue addressed along each research axis.

This positioning makes the novelty claims local rather than universal. The thesis does not claim to replace every robust-loss, weighting, or uncertainty-quantification method. Its contribution is a coordinated set of analytically motivated mechanisms that target specific combinations of properties within hyperplane-based and randomized neural models while retaining computationally simple model structures.

Table 2.6: Comparative positioning of the thesis within the relevant learning literature.

Research axis	**Representative literature**	**Unresolved limitation**	**Position of this thesis**
Robust classification losses	Margin losses; focal, generalized cross-entropy, symmetric cross-entropy, and bi-tempered losses	Boundedness, smoothness, sparsity, and asymmetric or noise-insensitive behavior rarely occur together; probability and margin formulations address different interfaces	Analytically designed bounded smooth margin losses, with RoBoSS emphasizing sparsity and Wave emphasizing asymmetric controlled influence
Robust regression losses	Huber, insensitive, canal, capped least-squares, BLINEX, and redescending M-estimators	Existing smooth bounded losses generally lack an exact insensitive zone; insensitive losses generally introduce non-smooth transition points	A smooth bounded regression loss with an explicit zero-loss interval, embedded in LSSVR
Imbalance-aware weighting	Global class weights, center/radius-based fuzzy memberships, and local-neighborhood probabilities	A single geometric or probabilistic signal may confuse boundary samples, multimodal structure, and noise; tuning and high-dimensional neighborhoods remain difficult	A weighting mechanism combining radius-adaptive proximity, local consistency, and class-ratio correction, with its practical scope stated explicitly
Randomized output learning	RVFL and BLS with squared-error closed-form output learning; fuzzy and robust variants	Squared error is outlier-sensitive and direct inversion can dominate computational cost; iterative alternatives have convergence-dependent cost	Bounded-loss randomized learning with inversion-free gradient optimization and an explicit conditional scalability analysis
Randomized initialization	Independent Gaussian or uniform hidden weights	Independent sampling ignores empirical feature dependence and may increase variability across random representations	Copula-aligned initialization that transfers estimated feature dependence into frozen hidden weights

Chapter 3

RoBoSS: A Robust, Bounded, Sparse, and Smooth Loss Function for Supervised Learning

This chapter develops the robust, bounded, sparse, and smooth (RoBoSS) loss function for supervised classification and studies its impact in both support vector machine (SVM) and twin support vector machine (TSVM) frameworks. The motivation is that widely used loss functions often struggle with outlier-prone, noisy, and high-dimensional data, leading to reduced robustness, slow convergence, or unstable optimization. The first part of the chapter introduces RoBoSS as a smooth bounded loss with sparsity-inducing behavior, establishes its classification-calibrated property and generalization ability, and embeds it into SVM to obtain the $\mathcal{L}_{RoBoSS}$-SVM model. The second part extends the same loss-design principle to TSVM and develops the RoBoSS-based Twin Support Vector Machine (RoBoTS), together with an inverse-free variant that avoids matrix inversion while preserving robustness and scalability. The chapter evaluates these models on benchmark, noisy, outlier-contaminated, and biomedical datasets, including EEG, BreaKHis, and ADNI, and presents statistical evidence that RoBoSS-based formulations improve classification performance, training effi-

ciency, and reliability in challenging real-world settings.

3.1 Introduction and Motivation

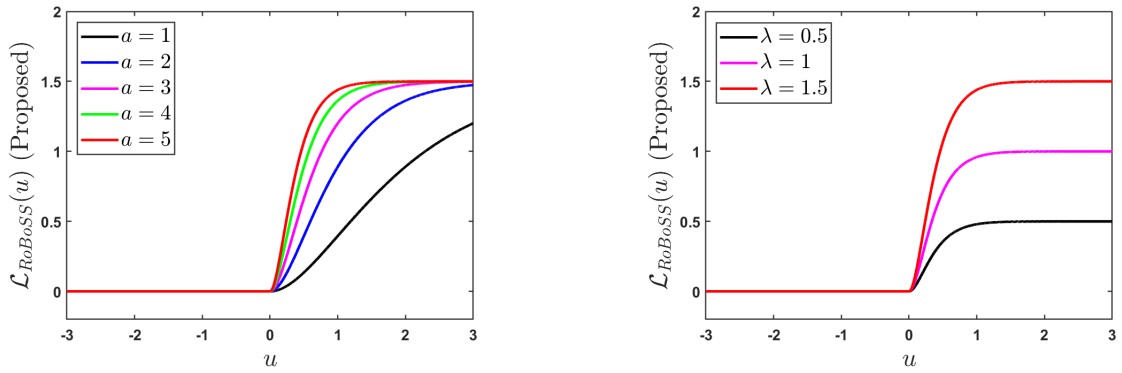
Data analysis tasks such as classification and regression fall under the umbrella of supervised machine learning (SML), where a model learns from labeled observations to make predictions on unseen instances. The loss function is central to this process because it determines how each training sample contributes to the empirical risk. In this chapter, the effect of loss design is first studied in the SVM framework and then extended to TSVM.

For binary classification, let the training set be $\{(x_k, y_k)\}_{k=1}^{n}$, where $x_k \in \mathbb{R}^m$ and $y_k \in \{1, -1\}$. In this opening SVM formulation only, n denotes the number of training samples and m denotes the feature dimension; this local convention is retained to match the original RoBoSS-SVM derivation. A generic soft-margin SVM objective can be written as

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + \frac{\mathcal{C}}{n}\sum_{k=1}^{n}\mathcal{L}(1 - y_k\left(w^\intercal x_k + b\right)), \tag{3.1}$$

where $\mathcal{C} > 0$ is a trade-off parameter and $\mathcal{L}(u)$, with $u = 1 - y_k(w^\intercal x_k + b)$, denotes the classification loss. The relevant baseline losses and their properties are summarized in Table 2.2. The gap motivating this chapter is that sparse bounded losses are usually non-smooth, while smooth losses are often unbounded or non-sparse. Hence, robust SVM learning requires a loss that jointly provides boundedness, sparsity, and smoothness.

Motivated by these observations, the main focus of this chapter is to construct a new robust, bounded, sparse, and smooth loss function for supervised classification. The RoBoSS



(a) (b)

Figure 3.1: RoBoSS loss behavior: (a) fixed $\lambda = 1.5$ with different values of a, and (b) fixed $a = 5$ with different values of λ.

loss function (see Fig. 3.1a and Fig. 3.1b) is defined as

$$\mathcal{L}_{RoBoSS}(u) = \begin{cases} \lambda\{1 - (au + 1)\exp(-au)\}, & u > 0, \\ 0, & u \leq 0, \end{cases} \tag{3.2}$$

where $a, \lambda > 0$ represent the shape and bound parameters, respectively. Further, we embed the proposed RoBoSS loss into SVM and introduce a new robust SVM model termed $\mathcal{L}_{RoBoSS}$-SVM. Replacing $\mathcal{L}(\cdot)$ by $\mathcal{L}_{RoBoSS}(\cdot)$ in (3.1) yields

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + \frac{\mathcal{C}}{n}\sum_{k=1}^{n}\mathcal{L}_{RoBoSS}\Big(1 - y_k\left(w^\intercal x_k + b\right)\Big). \tag{3.3}$$

The non-convex nature of the proposed loss function poses challenges for optimizing the $\mathcal{L}_{RoBoSS}$-SVM by the Wolfe-dual method. However, the smooth nature of $\mathcal{L}_{RoBoSS}$-SVM enables the use of gradient-based fast optimization techniques for solving the model. In this chapter, we utilize the Nestrov accelerated gradient (NAG) based framework to solve the optimization problem of $\mathcal{L}_{RoBoSS}$-SVM. NAG is known for its low computational complexity and efficiency in handling large-scale problems [65]. The main contributions of this work can be summarized as follows:

- We introduce an innovative advancement in the field of supervised learning: the RoBoSS (Robust, Bounded, Sparse, and Smooth) loss function.

- We explored the theoretical aspects of the RoBoSS loss and showed it possesses two crucial properties: classification-calibration and a bound on generalization error. These results not only emphasize the robustness of the RoBoSS loss function but also provide valuable insights into its performance and applicability.

- We fuse the RoBoSS loss within the SVM framework and introduce a novel SVM model coined as $\mathcal{L}_{RoBoSS}$-SVM. The resulting $\mathcal{L}_{RoBoSS}$-SVM model harnesses the inherent strengths of both the RoBoSS loss function and the SVM algorithm, leading to an advanced and versatile machine learning tool.

- We carried out numerical experiments on 88 benchmark KEEL and UCI datasets from diverse domains. The outcomes validate the effectiveness of the $\mathcal{L}_{RoBoSS}$-SVM model when compared to the baseline models.

- Furthermore, to showcase the prowess of the $\mathcal{L}_{RoBoSS}$-SVM in the biomedical domain, we executed additional evaluations on two biomedical datasets: the electroencephalogram (EEG) signal dataset and the breast cancer (BreaKHis) dataset. These experiments provide evidence of the model's efficiency in the biomedical realm.

3.2 RoBoSS Loss and RoBoSS-SVM Formulation

In this work, we present a significant advancement in supervised learning: a new loss function characterized by robustness, boundedness, sparsity, and smoothness, termed the RoBoSS loss (see Fig. 3.1a and 3.1b). This innovative approach represents a substantial stride in optimizing the training process of machine learning models. The equation (3.2)

provides the mathematical formulation of the RoBoSS loss introduced in this study. The RoBoSS loss function, as put forth in this work, exhibits the subsequent characteristics:

- It is robust and sparse. By setting an upper bound λ and capping the loss for samples with $u > 0$ beyond a certain margin, robustness is enhanced. Additionally, it assigns a fixed loss of 0 for all samples with $u \leq 0$, thereby introducing sparsity.
- It is smooth, bounded, and non-convex.
- It has two beneficial parameters, a and λ, known as the shape parameter and bounding parameter, respectively. The shape parameter (a) controls the intensity of the penalty, while the bounding parameter (λ) defines the limits for the loss values.
- For $\lambda = 1$, when $a \to +\infty$, it approaches the "$0-1$" loss function in a pointwise manner.

The RoBoSS loss function addresses multiple crucial aspects of supervised learning simultaneously. By encompassing robustness, it ensures the stability of the learning process even in the presence of outliers. The bounded nature of the RoBoSS loss function restricts the impact of extreme values, preventing the loss from growing unbounded. Incorporating sparsity, the RoBoSS loss function promotes the utilization of only the samples that are misclassified or near the decision boundary, resulting in parsimonious models. Moreover, the RoBoSS loss function is designed with a focus on smoothness, facilitating a gradual and consistent optimization process. This smoothness property promotes avoiding abrupt changes during parameter updates, leading to more stable and efficient convergence during training. Next, to highlight the advantages of the proposed RoBoSS loss function, we provide a thorough comparison with existing loss functions:

1. **Robustness to outliers:** Traditional loss functions, including the hinge loss, pinball loss, and LINEX loss, are both unbounded and convex. Although convexity provides

certain advantages, the unbounded nature of these functions renders them highly sensitive to outliers. Conversely, the RoBoSS loss function is bounded, greatly improving its robustness to outliers. The bounding parameter λ ensures that the loss value does not increase indefinitely for any sample, thereby preventing outliers from disproportionately influencing the model training. This prevents the model from being unduly influenced by extreme values, ensuring a more balanced learning process.

2. **Flexibility in penalty assignment:** Existing loss functions like the hinge loss and pinball loss, and their variants do not possess a shape parameter and assign a uniform loss value to misclassified samples, regardless of the dataset's characteristics. This uniform penalty approach can be suboptimal when dealing with diverse data distributions, as it does not allow for adjustments based on the specific requirements of different datasets. The RoBoSS loss function offers flexibility in managing different data distributions through its shape parameter a. This parameter allows for tuning the strength of the penalty assigned to misclassified samples. By adjusting a, one can control the severity of the penalization for misclassifications, providing the ability to adapt the loss function to various data characteristics (see Fig. 3.1a). This flexibility is particularly advantageous when dealing with heterogeneous datasets, as it enables the model to be more responsive to the specific needs of different data distributions.

3. **Sparsity:** Sparse models are often easier to analyze because they rely on fewer support vectors [66]. The hinge loss, while sparse, lacks boundedness and can lead to suboptimal results on outlier-prone datasets. Pinball loss and LINEX loss, on the other hand, sacrifice both sparsity and boundedness. Wave loss [67], while bounded, lacks the sparsity property. However, the proposed RoBoSS loss strikes a balance by being both sparse and bounded. It enhances sparsity by assigning zero loss to all

correctly classified samples. This characteristic ensures that only the most relevant samples contribute to the model's training, leading to simpler models.

4. **Smoothness:** The non-smooth nature of traditional loss functions such as hinge loss, pinball loss, truncated hinge loss, and truncated pinball loss can lead to challenges in optimization, often requiring specialized algorithms for convergence. The proposed RoBoSS loss function, with its inherent smoothness, allows for the use of gradient-based fast optimization techniques. This smoothness avoids abrupt changes during parameter updates, promoting a more consistent optimization trajectory. The smooth nature of RoBoSS ensures stable updates during training, leading to efficient and effective model convergence.

To succinctly illustrate the advantages of the proposed RoBoSS loss function, Table 3.1 compares the key characteristics of various state-of-the-art loss functions with RoBoSS.

Table 3.1: Illustrates the key attributes of different state-of-the-art loss functions with the proposed RoBoSS loss function, highlighting their robustness, sparsity, boundedness, convexity, and smoothness.

Loss function $\downarrow\backslash$ Characteristic $\rightarrow$	**Robust to outliers**	**Sparse**	**Bounded**	**Convex**	**Smooth**
Hinge loss [6]	✗	✓	✗	✓	✗
Pinball loss [27]	✗	✗	✗	✓	✗
Truncated hinge loss [68]	✓	✓	✓	✗	✗
Truncated pinball loss [36]	✓	✗	✓	✗	✗
LINEX loss [31]	✗	✗	✗	✓	✓
QTSELF loss [69]	✗	✗	✗	✗	✓
Wave loss [67]	✓	✗	✓	✗	✓
RoBoSS loss (Proposed)	✓	✓	✓	✗	✓

Now, by amalgamating the RoBoSS loss function (3.2) within the least squares SVM framework, we introduce a novel SVM model termed $\mathcal{L}_{RoBoSS}$-SVM. For simplicity, we adopt the notation w to represent $[w^\intercal, b]$ and x_i to represent $[x_i, 1]^\intercal$, henceforth. The

$\mathcal{L}_{RoBoSS}$-SVM model is delineated as follows:

$$\begin{aligned} \min_{w,\xi} \ & \frac{1}{2}\|w\|^2 + \frac{\mathcal{C}}{n}\sum_{k=1}^{n}\lambda\Big(1-(a\{\xi_k\}_+ + 1)exp(-a\{\xi_k\}_+)\Big), \\ \text{s.t. } & y_k\left(w^\intercal\psi(x_k)\right) = 1-\xi_k, \ \forall\, k = 1,2,\ldots,n, \end{aligned} \tag{3.4}$$

where $\{\xi_k\}_+ = \xi_k$ if $\xi_k > 0$ and 0 otherwise, $\mathcal{C} > 0$ is the regularization parameter, a and λ are the loss parameters, and the function $\psi(\cdot)$ represents the feature mapping corresponding to the kernel function.

While kernel functions are typically used to handle non-linear problems through dual problem formulation, the non-convex nature of $\mathcal{L}_{RoBoSS}$-SVM makes this approach formidable. To empower the non-linear adaptation capability of $\mathcal{L}_{RoBoSS}$-SVM, we utilize the representer theorem [70]. Using the representer theorem [70], the corresponding solution can be stated as:

$$w = \sum_{k=1}^{n}\beta_k\psi(x_k), \tag{3.5}$$

where $\beta = (\beta_1,\ldots,\beta_n)^\intercal$ represents the coefficient vector. By substituting the value of w from equation (3.5) into equation (3.4), we derive

$$\begin{aligned} \min_{\beta} \ f(\beta) = & \sum_{k=1}^{n}\sum_{j=1}^{n}\frac{1}{2}\beta_k\beta_j\mathcal{K}\left(x_k,x_j\right) \\ & + \frac{\mathcal{C}}{n}\sum_{k=1}^{n}\lambda\Big(1-(a\{\xi_k\}_+ + 1)exp(-a\{\xi_k\}_+)\Big), \end{aligned} \tag{3.6}$$

where $\xi_k = y_k\left(\sum_{j=1}^{n}\beta_j\mathcal{K}\left(x_k,x_j\right)\right) - 1$, and $\mathcal{K}\left(x_k,x_j\right) = \psi\left(x_k\right)\cdot\psi\left(x_j\right)$ is the kernel function.

3.3 Theoretical Analysis of the RoBoSS Loss

Assume that the training data $z = \{x_k, y_k\}_{k=1}^{n}$ is drawn independently from a probability measure $\mathcal{P}$. The probability measure $\mathcal{P}$ is defined on $X \times Y$, where $X \subseteq \mathbb{R}^m$ represents the input space and $Y = \{-1, 1\}$ is the label space. The primary objective of the classification task is to produce a function $\mathcal{C} : X \to Y$ that reduces the related risks. The risk related with $\mathcal{C}$ is defined as follows:

$$\mathcal{R}(\mathcal{C}) = \int_X \mathcal{P}(y \neq \mathcal{C}(x)|x) d\mathcal{P}_X,$$

where $\mathcal{P}(y|x)$ represents the conditional probability distribution of $\mathcal{P}$ given x and $d\mathcal{P}_X$ is the marginal distribution of $\mathcal{P}$ on x. Further, $\mathcal{P}(y|x)$ adheres to a binary distribution, expresses as the likelihoods $\text{Prob}(y = 1|x)$ and $\text{Prob}(y = -1|x)$. To simplify, we denote $\text{Prob}(y = 1|x)$ as $P(x)$ and $\text{Prob}(y = -1|x)$ as $1 - P(x)$. Now, for $P(x) \neq 1/2$, the Bayes classifier function ($f_{\mathcal{C}}(x)$) assigns a value of 1 if $P(x) > 1/2$ and -1 if $P(x) < 1/2$. It can be demonstrated that the Bayes classifier achieves the minimum classification risk [27]. Practically, we aim to identify a function $f{:}X \to \mathbb{R}$ that can generate a binary classifier. In this case, the classification risk becomes $\int_{X \times Y} \mathcal{L}_{mis}(yf(x)) d\mathcal{P}$, where $\mathcal{L}_{mis}(yf(x))$ is the misclassification loss defined as

$$\mathcal{L}_{mis}(yf(x)) = \begin{cases} 0, & yf(x) > 0, \\ 1, & yf(x) \leq 0. \end{cases} \tag{3.7}$$

Therefore, minimizing the misclassification error will result in a function whose sign corresponds to the Bayes classifier [27]. Now, the expected risk of a classifier $f : X \to \mathbb{R}$ for any given loss function $\mathcal{L}$ can be expressed as:

$$\mathcal{R}_{\mathcal{L},\mathcal{P}}(f) = \int_{X \times Y} \mathcal{L}(1 - yf(x)) d\mathcal{P}. \tag{3.8}$$

The function $f_{L,P}$, which achieves the lowest expected risk among all measurable functions, can be described as follows:

$$f_{\mathcal{L},\mathcal{P}}(x) = \arg\min_{f(x)\in\mathbb{R}} \int_Y \mathcal{L}\left(1 - yf(x)\right) d\mathcal{P}(y|x),\ \forall x \in X. \tag{3.9}$$

Then, for the RoBoSS loss ($\mathcal{L}_{RoBoSS}(\cdot)$), we can obtain Theorem 3.3.1, demonstrating that the RoBoSS loss is classification-calibrated [26]. It is a desirable property for a loss function and requires that the minimizer of the function $\mathcal{R}_{\mathcal{L},\mathcal{P}}(f)$ shares the sign as of the Bayes classifier. Classification calibration, as introduced by Bartlett et al. [26], provides a framework for evaluating the statistical efficacy of a loss function. It ensures that the probabilities predicted by the model are closely aligned with the true event probabilities, thereby enhancing the fidelity of the model's predictions.

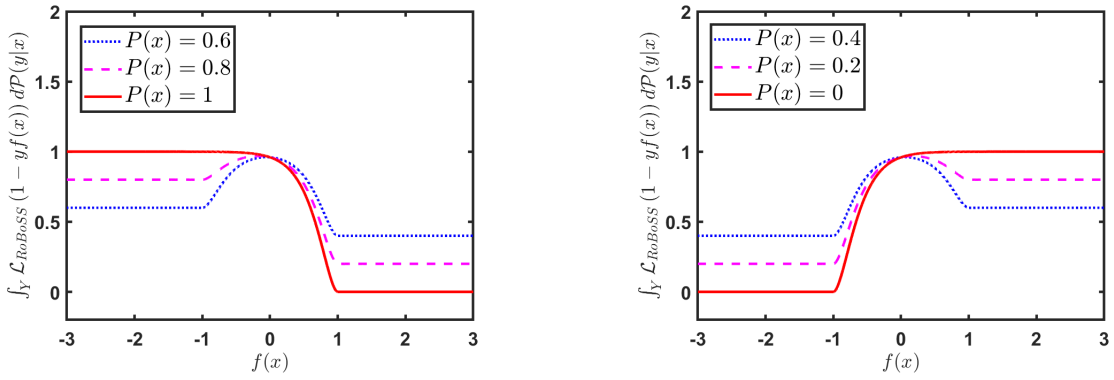
Theorem 3.3.1. *The proposed loss $\mathcal{L}_{RoBoSS}(u)$ is classification-calibrated, i.e., $f_{\mathcal{L}_{RoBoSS},\mathcal{P}}$ has the same sign as the Bayes classifier.*

Proof. After simple calculation, we obtain that

$$\begin{aligned}
&\int_Y \mathcal{L}_{RoBoSS}\left(1 - yf(x)\right) d\mathcal{P}(y|x) \\
&= \mathcal{L}_{RoBoSS}(1 - f(x))P(x) + \mathcal{L}_{RoBoSS}(1 + f(x))(1 - P(x)) \\
&= \begin{cases} g_1(x)P(x), & f(x) \le -1, \\ (g_1(x) - g_2(x))P(x) + g_2(x), & -1 < f(x) < 1, \\ g_2(x)(1 - P(x)), & f(x) \ge 1, \end{cases}
\end{aligned}$$

where $g_1(x) = \lambda\Big(1 - (a(1 - f(x)) + 1)exp(-a(1 - f(x)))\Big)$ and $g_2(x) = \lambda\Big(1 - (a(1 + f(x)) + 1)exp(-a(1 + f(x)))\Big)$.

Fig. 3.2a and 3.2b show the graph of $\int_Y \mathcal{L}_{RoBoSS}\left(1 - yf(x)\right) d\mathcal{P}(y|x)$ over $f(x)$ when



(a) (b)

Figure 3.2: Demonstrate the graph of $\int_Y \mathcal{L}_{RoBoSS}\,(1 - yf(x))\,d\mathcal{P}(y|x)$ with respect to $f(x)$ for different $P(x)$ values. (a) For $P(x) > 1/2$ and (b) for $P(x) < 1/2$.

$P(x) > 1/2$ and $P(x) < 1/2$, respectively. It is evident from Fig. 3.2 that, for $P(x) > 1/2$, the minimum value of $\int_Y \mathcal{L}_{RoBoSS}\,(1 - yf(x))\,d\mathcal{P}(y|x)$ is obtained for the positive value of $f(x)$, and for $P(x) < 1/2$, it is obtained for the negative value of $f(x)$. Therefore, we can conclude that the function corresponding to the RoBoSS loss, which minimizes the expected risk overall measurable functions, has the same sign as the Bayes classifier.

Hence, the proposed loss $\mathcal{L}_{RoBoSS}(u)$ is classification-calibrated. □

Further, we investigate the generalization ability of $\mathcal{L}_{RoBoSS}$-SVM. First, we define the Rademacher complexity, which measures the complexity of a class of functions.

Definition 3.3.1. Rademacher Complexity [71]

Let $\mathcal{X}$:=$\{x_1, x_2, \ldots, x_p\}$ be drawn independently from $d\mathcal{P}_X$ and $\mathcal{G}$ be a class of functions from X to $\mathbb{R}$. Define the random variable

$$\hat{R}_p(\mathcal{G}) := \mathbb{E}\left[\sup_{g\in\mathcal{G}}\left|\frac{2}{p}\sum_{k=1}^{p}\theta_k g\,(x_k)\right| \mid \mathcal{X}\right],$$

where $\theta_1, \theta_2, \ldots, \theta_p$ are independent discrete uniform $\{\pm 1\}$-valued random variables. Then the Rademacher complexity of $\mathcal{G}$ is $R_p(\mathcal{G}) = \mathbb{E}\hat{R}_p(\mathcal{G})$.

Now, let the expected risk and empirical risk of RoBoSS loss be denoted by $\mathcal{R}(f_c)$ and $\mathcal{R}_z(f_c)$, respectively, and defined as

$$\mathcal{R}(f_c) = \int_{X\times Y} \mathcal{L}_{RoBoSS}(1 - yf(x))d\mathcal{P},$$

$$\mathcal{R}_z(f_c) = \frac{1}{n}\sum_{k=1}^{n} \mathcal{L}_{RoBoSS}(1 - yf(x)).$$

Then the generalization ability of $\mathcal{L}_{RoBoSS}$-SVM can be stated as the convergence of $\mathcal{R}_z(f_c)$ to $\mathcal{R}(f_c)$ when the sample size n tends to infinity, where f_c is the classifier elicited by (3.4).

Theorem 3.3.2. *Let f_c be the classifier produced by $\mathcal{L}_{RoBoSS}$-SVM. Then for any $0 < \varepsilon < 1$, with confidence $1 - \varepsilon$, the following inequality holds*

$$\mathcal{R}(f_c) - \mathcal{R}_z(f_c) \leq \frac{4\lambda}{\sqrt{n\mathcal{C}}} + \sqrt{\frac{8\ln(1/\varepsilon)}{n}}.$$

Proof. For classifier f_c, obtained by (3.4) with the regularization parameter $\mathcal{C}$, we have

$$\mathcal{C}\left\|f_c^{\mathcal{L}_{RoBoSS}}\right\|_{\mathcal{K}}^2 \leq \lambda^2,$$

which implies $\left\|f_c^{\mathcal{L}_{RoBoSS}}\right\|_{\mathcal{K}} \leq \lambda/\sqrt{\mathcal{C}}$ [72]. Now, using Theorem 8 of [71], for any $0 < \varepsilon < 1$, we have

$$\mathcal{R}\left(f_c^{\mathcal{L}_{RoBoSS}}\right) - \mathcal{R}_{\mathbf{z}}\left(f_c^{\mathcal{L}_{RoBoSS}}\right) \leq R_n(\mathcal{J}) + \sqrt{\frac{8\ln(1/\varepsilon)}{n}}, \tag{3.10}$$

where the set $\mathcal{J}$ is defined as

$$\mathcal{J} := \Big\{ j \mid j(x,y) := \phi(1 - yf(x)) - \phi(0), f \in \mathcal{J}_{\mathcal{K}}, \\ \|f\|_{\mathcal{K}} \leq \lambda/\sqrt{\mathcal{C}}, (x,y) \in X \times Y \Big\}.$$

Again, Theorem 12 of [71] yields that

$$R_n(\mathcal{J}) \leq 2R_n\left(\mathcal{G}_{\mathcal{C}}\right) \text{ with}$$
$$\mathcal{G}_{\mathcal{C}} = \left\{ f \mid f \in \mathcal{J}_{\mathcal{K}}, \|f\|_{\mathcal{K}} \leq \lambda\sqrt{\log\left(1+\lambda^{-2}\right)/\mathcal{C}} \right\}.$$

Also from [73], we have

$$R_n\left(\mathcal{G}_{\mathcal{C}}\right) \leq \frac{2\lambda}{\sqrt{n\mathcal{C}}}. \tag{3.11}$$

Hence, from (3.10) and (3.11), for any $0 < \varepsilon < 1$, we have

$$\mathcal{R}(f_c^{\mathcal{L}_{RoBoSS}}) - \mathcal{R}_z(f_c^{\mathcal{L}_{RoBoSS}}) \leq \frac{4\lambda}{\sqrt{n\mathcal{C}}} + \sqrt{\frac{8\ln(1/\varepsilon)}{n}}.$$

□

3.4 Optimization of RoBoSS-SVM

To solve the optimization problem (3.6), we adopt the framework based on the Nestrov accelerated gradient (NAG) algorithm. It is an extension of the stochastic gradient descent (SGD) method that incorporates momentum to accelerate convergence. In SGD, a small batch of samples (mini-batch) is used for each iteration during the training of a model. This approach offers several advantages, including reduced computational requirements and improved speed, particularly when dealing with large-scale problems. However, SGD has some drawbacks, such as getting stuck in local optima during its process of convergence due to the randomness of the mini-batch. To improve SGD, many researchers introduced accelerated variance in SGD [74, 75]. The momentum method [76] is a practical approach that helps SGD to accelerate in the relevant direction and dampen the oscillation. It does this by combining the update vector of the previous time step with the current update vector.

The NAG algorithm is an extension of the momentum method that further improves convergence by incorporating a "look-ahead" mechanism [61]. It gives an approximation of the future position of the parameters and then calculates the gradient with respect to the approximate future position of the model parameters. One challenge for NAG is to choose an appropriate learning rate during the training. If the learning rate is set to a very low value, the algorithm's convergence speed becomes sluggish. On the contrary, using a high learning rate is likely to cause the algorithm to overshoot the optimal point or even fail to converge. An intuitive approach is to begin with a slightly higher learning rate and then gradually reduce it during the learning process according to a predefined schedule. Taking inspiration from the simulated annealing approach [77], we employ the exponential decay method for adjusting the learning rate as $\alpha_{new} = \alpha_{old} \exp(-\eta t)$, where η is a hyperparameter that determines the extent of the learning rate's decay at each iteration, while t represents the current iteration number.

Now, we solve (3.6) by employing the NAG-based framework. The method employed to solve (3.6) is thoroughly described in Algorithm 3.1. After obtaining the optimal β, the subsequent decision function is used to classify a test sample $\hat{x}$:

$$\hat{y} = \text{sign}\left(\sum_{j=1}^{s} \beta_j \mathcal{K}\left(x_j, \hat{x}\right)\right). \tag{3.12}$$

To elucidate the integration of RoBoSS loss into the SVM framework, Fig. 3.3 presents a flowchart of the proposed $\mathcal{L}_{RoBoSS}$-SVM model that encapsulates our methodology. This visual representation serves as both a guide to our methodology and a demonstration of the strategic integration of its components.

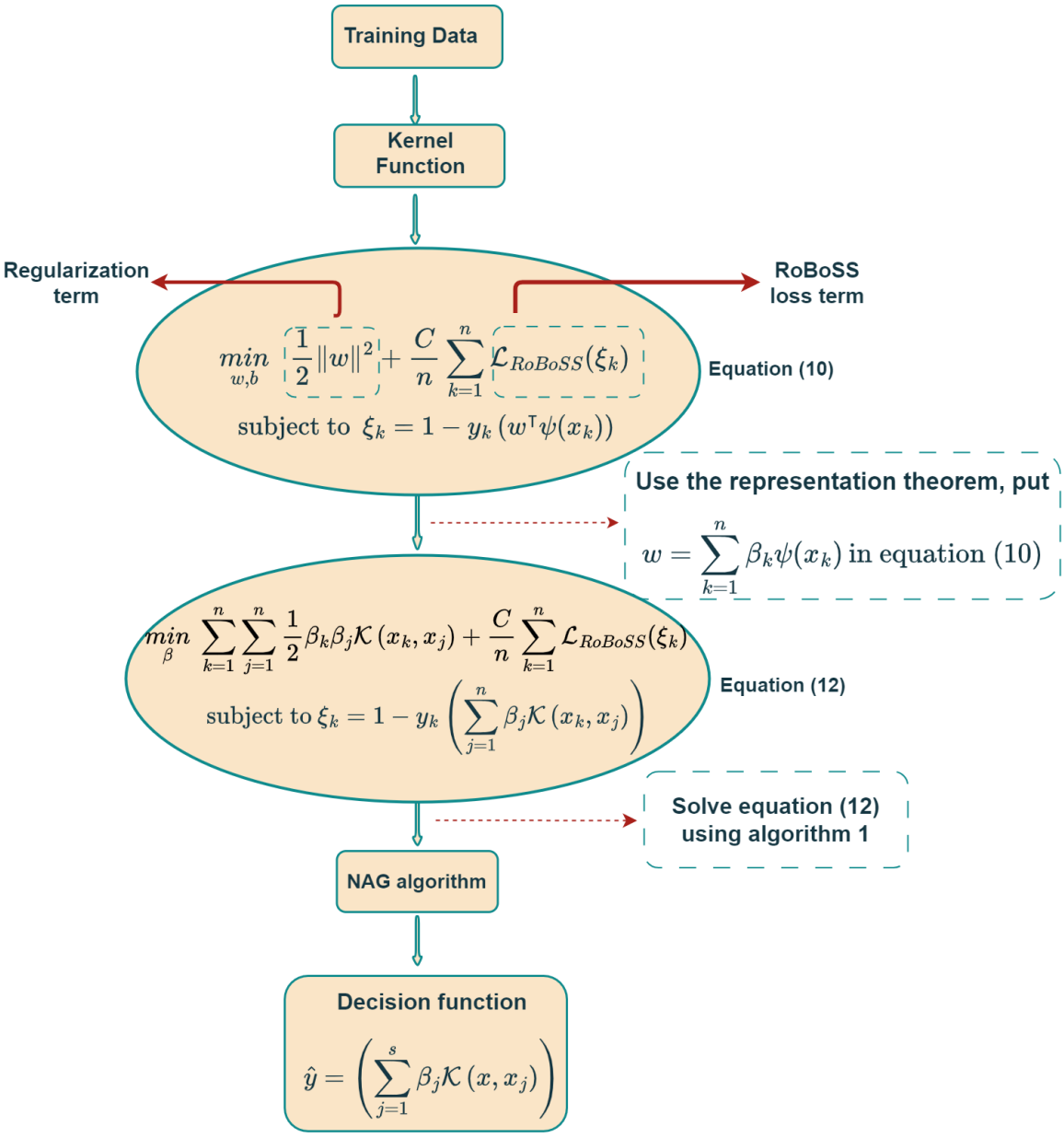


Figure 3.3: Flowchart of the proposed $\mathcal{L}_{RoBoSS}$-SVM model. It depicts the essential stages of the proposed $\mathcal{L}_{RoBoSS}$-SVM, demonstrating the process from raw input data to the final decision function. This flowchart shows the incorporation of the RoBoSS loss function and illustrates the use of the representer theorem and the NAG optimization algorithm.

Algorithm 3.1 NAG-based algorithm to solve $\mathcal{L}_{RoBoSS}$-SVM

Input:

The dataset: $\{x_k, y_k\}_{k=1}^{n}$, $y_k \in \{-1, 1\}$;

The parameters: Regularization parameter $\mathcal{C}$, RoBoSS loss parameters λ and a, mini-batch size s, learning rate decay factor η, momentum parameter r, maximum iteration number N;

Initialize: model parameter β_0, velocity υ_0, learning rate α; **Output:**

The classifiers parameters: β;

1: Select s samples $\{x_k, y_k\}_{k=1}^{s}$ uniformly at random.

2: Computing ξ_k :

$$\xi_k = 1 - y_k \left(\sum_{j=1}^{s} \beta_j \mathcal{K}\left(x_k, x_j\right) \right), \; k = 1, \cdots, s; \tag{3.13}$$

3: Temporary update: $\widetilde{\beta}_t = \beta_t + r v_t$;

4: Compute gradient:

$$\nabla f(\widetilde{\beta}_t) = \mathcal{K}\widetilde{\beta}_t - \frac{\mathcal{C}}{s}\lambda \sum_{j=1}^{s} a^2 \xi_j \exp\left(-a\xi_j\right) y_j \mathcal{K}_j, \tag{3.14}$$

where $\mathcal{K}$ denotes the kernel matrix and $\mathcal{K}_j$ denotes the jth row of $\mathcal{K}$.

5: Update velocity: $v_t = r v_{t-1} - \alpha_{t-1} \nabla f(\widetilde{\beta}_t)$;

6: Update model parameter: $\widetilde{\beta_{t+1}} = \widetilde{\beta}_t + v_t$;

7: Update learning rate: $\alpha_{t+1} = \alpha_t \exp(-\eta t)$;

8: Update current iteration number: $t = t + 1$.

Until:

$t = N$.

Return: β_t.

3.4.1 Computational Analysis

In this subsection, we provide an analysis of the computational complexity of the NAG algorithm (Algorithm 3.1) utilized to solve the optimization problem of the proposed $\mathcal{L}_{RoBoSS}$-SVM. Consider l and m represent the sample and feature counts, respectively, in the training data, while N indicates the count of iterations. The computational complexity associated with updating the variable v can be expressed as $\mathcal{O}\left((N+m)l^2\right)$. Similarly, the complexity for updating the parameter β is $\mathcal{O}(Nl)$, lastly the complexity of updating the parameter α is $O(N)$ [49]. Combining these complexities, the total computational complexity for the NAG algorithm in solving the $\mathcal{L}_{RoBoSS}$-SVM can be summarized as $\mathcal{O}\left((N+m)l^2\right)$. It is important to note that the computational complexity of the traditional SVM model is $\mathcal{O}(l^3)$. This highlights that the proposed $\mathcal{L}_{RoBoSS}$-SVM, with its computational complexity of $\mathcal{O}\left((N+m)l^2\right)$, offers a significant improvement in efficiency over the traditional SVM model, particularly as the number of samples l increases. This enhanced efficiency underscores the practicality and scalability of the $\mathcal{L}_{RoBoSS}$-SVM for large-scale datasets.

3.5 RoBoSS-SVM Experimental Evaluation

This section gives the experimental protocol, benchmark evaluation, noise and outlier studies, biomedical validation, statistical tests, and sensitivity analysis for RoBoSS-SVM. The extended experimental material is included here so that the chapter is self-contained.

3.5.1 Experimental Setup and Parameter Selection

The experiments are conducted using MATLAB R2023a on a Windows 10 system equipped with an Intel(R) Xenon(R) Platinum 8260 CPU running at 2.30 GHz and 256 GB of RAM. To map the input samples into a higher-dimensional space, the Gaussian ker-

nel function is used. It is defined as $\kappa(x_k, x_j) = \exp\left(-\|x_k - x_j\|^2/\sigma^2\right)$, where σ is the kernel parameter. Before training, each dataset is normalized in the interval $[-1, 1]$. For each model, the penalty parameter $\mathcal{C}$ and the kernel parameter σ are chosen from the range $\{10^i \mid i = -6 : 1 : 6\}$. For $\mathcal{L}_{pin}$-SVM, the hyperparameter τ is selected from $\{0, 0.3, 0.5, 0.7, 0.9\}$. For $\mathcal{L}_{LINEX}$-SVM, and $\mathcal{L}_{qtse}$-SVM the range of loss hyperparameter is taken the same as in [31] and [69], respectively. For $\mathcal{L}_{wave}$-SVM, the loss hyperparameter a is selected from the range $[-5 : 1 : 5]$ and the bounding parameter λ is fixed to 1. For the proposed $\mathcal{L}_{RoBoSS}$-SVM, the loss parameters a and λ are chosen from the range $[0 : 0.1 : 5]$ and $[0.1 : 0.1 : 2]$, respectively. The parameters for the NAG-based algorithm are experimentally set as: (i) initial model parameter $\beta_0 = 0.01$, (ii) initial velocity $v_0 = 0.01$, (iii) initial learning rate $\alpha = 0.1$, (iv) learning decay factor $\eta = 0.1$, (v) momentum parameter $r = 0.6$, (vi) two distinct minibatch size configurations are used based on the size of the dataset: $s = 2^2$ for datasets with less than 100 samples and $s = 2^5$ for datasets with 100 or more samples, (vii) maximum iteration number $N = 1000$.

The choice of hyperparameters has a significant impact on the models' performance. In order to optimize them, we employ 5-fold cross-validation along with grid search. In this, the dataset is randomly split into five disjoint subsets, where one subset serves as the test set and the remaining four are used for training. For each set of hyperparameters, we determined the testing accuracy for all five subsets separately. Then, for each hyperparameter set, we calculate the mean testing accuracy by taking the average of these five accuracy values. The best mean testing accuracy is chosen as the testing accuracy of the model. The same folds, parameter-search procedure, preprocessing, and performance criterion are applied to every baseline and proposed model, so the tabulated values provide a procedurally controlled comparison under a common experimental setup. Nevertheless, the held-out folds used to choose the hyperparameters also contribute to the reported mean accuracy. This

protocol is therefore susceptible to model-selection bias and can yield a more optimistic estimate than nested cross-validation or evaluation on a test set that remains independent of parameter selection [78]. The reported margins should consequently be interpreted as comparative results under the stated common protocol, rather than as unbiased estimates of performance on a new external sample.

The performance of the models is evaluated using the accuracy metric, which is defined as

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \times 100,$$

where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively. To further analyze the model's performance, we also evaluate the rank and training time. The reported times reflect only the duration required to train the models using the best hyperparameters. The methods developed in this chapter primarily address robustness to label noise and outliers through loss-function design and are not specifically formulated for class-imbalance-aware learning. Accordingly, accuracy and average rank are used as the principal measures for comparing all baseline and proposed models under identical data partitions and experimental settings. In contrast, Chapter 6 explicitly addresses class imbalance through an adaptive weighting scheme and therefore reports F-measure alongside accuracy.

For the imbalanced KEEL datasets, however, accuracy alone does not characterize minority-class sensitivity. The retained experimental outputs contain aggregate fold accuracies but not the fold-level predictions, decision scores, or confusion matrices required to compute F-measure, G-mean, or area under the receiver operating characteristic curve reliably. The corresponding tables therefore retain accuracy and rank, and their conclusions are restricted to overall predictive performance under the stated protocol. Future evalua-

tions should preserve fold-level outputs and report class-sensitive metrics within a nested or independently validated experimental design.

The detailed procedure for artificially introducing outliers and label noise into the training dataset is outlined as follows:

Methodology for Introducing Outliers:

Outliers have been systematically introduced into the datasets at varying levels: 5%, 10%, 20%, and 30% of the total number of samples. For each dataset, the number of outliers has been calculated based on the specified percentage. Random samples have then been selected to serve as outliers. For each chosen sample, a feature has been randomly selected, and its value has been altered by multiplying it by an outlier factor of 10. This systematic introduction of outliers allows us to rigorously test the robustness and stability of the $\mathcal{L}_{RoBoSS}$-SVM model in handling data contamination.

Methodology for Introducing Label Noise:

Label noise has been introduced by randomly flipping the labels of a certain percentage of the samples. Specifically, noise levels of 5%, 10%, 20%, and 30% have been used. For each dataset, the number of labels to be flipped has been determined based on the specified noise level. Random samples have been selected, and their labels have been inverted to create noise.

3.5.2 Evaluation on KEEL and UCI Datasets

Here, we present the experimental results on 88 real-world datasets downloaded from the KEEL [79] and UCI [80] repositories. Based on the sample size, we split the datasets into two categories: (D1) datasets with samples under or equal to 5000, and (D2) datasets with samples over 5000. There are 79 and 9 datasets in the D1 and D2 categories, respectively.

Tables 3.2 and 3.2 present the dataset-wise classification accuracies, training times, and

ranks for the 79 D1 category datasets.

Table 3.2: The average classification accuracies, training times, and ranks of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on each 79 D1 category UCI and KEEL datasets.

Model	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{gtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
Dataset samples, features	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time
acute inflammation 120,6	**100±0**, 0.0049	**100±0**, 0.0047	90±22.36, 0.0024	75±37.27, 0.0016	**100±0**, 0.0033	98.33±3.73, 0.001
balloons 16,4	53.33±50.55, 0.0079	80±18.26, 0.002	73.33±27.89, 0.0023	86.67±18.26, 0.0015	89.67±17.08, 0.0004	**100±0**, 0.0011
fertility 100,9	88±10.37, 0.0054	89±9.62, 0.0075	88±10.37, 0.0017	88±10.37, 0.0008	90±10.37, 0.0024	**92±7.58**, 0.0009
molec biol promoter 106,57	56.75±10.8, 0.0081	90.48±21.3, 0.0052	71.39±42.68, 0.0018	90.48±21.3, 0.0009	91.23±19.34, 0.0033	**92.38±17.04**, 0.0008
parkinsons 195,22	79.49±23.15, 0.0117	80±21.48, 0.0043	80±21.48, 0.0018	80±21.48, 0.0008	82.46±16.07, 0.0024	**89.74±9.59**, 0.0009
pittsburg bridges T OR D 102,7	86.14±13.95, 0.0078	86.14±13.95, 0.0029	86.14±13.95, 0.0028	86.14±13.95, 0.0008	86.14±13.95, 0.0028	**92.14±5.69**, 0.0008
breast cancer 286,9	70.18±44.62, 0.0047	70.18±44.62, 0.009	70.18±44.62, 0.0033	70.62±24.14, 0.0037	73.78±17.13, 0.0043	**81.81±12.26**, 0.0011
breast cancer wisc prog 198,33	76.35±8.95, 0.0036	76.35±8.95, 0.0057	77.35±8.32, 0.0047	76.35±8.95, 0.0016	76.35±8.95, 0.0025	**78.82±7.94**, 0.0011
congressional voting 435,16	62.07±3.04, 0.013	62.3±2.62, 0.0153	61.38±1.74, 0.0037	62.07±2.15, 0.0028	62.38±1.74, 0.0027	**63.22±3.25**, 0.0011
echocardiogram 131,10	75.44±11.83, 0.0042	77.01±7.85, 0.0051	77.04±7.35, 0.0036	75.58±11.04, 0.0017	77.81±8.01, 0.0027	**85.44±9.63**, 0.0011
haberman survival 306,3	73.49±8.48, 0.0046	73.49±8.48, 0.0088	73.82±8.2, 0.003	73.82±8.68, 0.003	73.49±8.48, 0.0028	**76.11±8.71**, 0.0011
hepatitis 155,19	83.23±11.27, 0.0039	83.87±7.21, 0.0035	81.94±8.1, 0.0036	81.29±11.5, 0.0017	82.58±11.08, 0.0028	**89.03±8.1**, 0.0011
horse colic 368,25	**80.17±4.3**, 0.0074	**80.17±4.3**, 0.0106	76.09±3.24, 0.0035	65.77±6.65, 0.0017	76.08±3.38, 0.0029	79.59±9.5, 0.0011
ionosphere 351,33	64.71±21.68, 0.0104	67.79±6.61, 0.028	**81.78±9.51**, 0.0045	69.61±23.45, 0.0021	71.35±5.09, 0.0045	72.56±4.84, 0.0012
planning 182,12	71.38±8.85, 0.0038	71.38±8.85, 0.0042	71.38±8.85, 0.0027	71.38±8.85, 0.0028	75.79±12.43, 0.0029	**86.91±8.61**, 0.0014
spect 265,22	64.15±6.67, 0.0043	65.28±6.62, 0.007	65.66±9.09, 0.0033	58.49±14.06, 0.0025	71.38±8.85, 0.0024	**73.6±3.89**, 0.0012
spectf 267,44	**79.34±20.89**, 0.0039	**79.34±20.89**, 0.0061	**79.34±20.89**, 0.0032	**79.34±20.89**, 0.0027	70.28±10.39, 0.0028	72.08±4.3, 0.0012
statlog heart 270,13	77.04±1.66, 0.0062	77.04±1.66, 0.0103	78.15±4.01, 0.0031	72.96±3.84, 0.0019	79.34±20.89, 0.0028	**80.48±10.33**, 0.0016
bupa or liver-disorders 345,6	71.88±3.64, 0.0074	71.88±3.64, 0.02	62.61±8.9, 0.0035	62.32±8.64, 0.003	76.3±7.1, 0.0027	**80.37±1.66**, 0.0011
cleve 297,13	**78.77±7.4**, 0.0078	**78.77±7.4**, 0.0165	76.07±5.26, 0.0036	74.36±8.95, 0.0026	64.93±6.51, 0.005	69.28±3.75, 0.0014
crossplane130 130,2	70.77±8.85, 0.004	70.77±8.85, 0.0057	70±8.34, 0.003	64.62±9.96, 0.0017	75.4±4.24, 0.0037	**80.79±4.34**, 0.0012
crossplane150 150,2	62±12.16, 0.0038	62±12.16, 0.0081	67.33±7.6, 0.0029	66.67±8.82, 0.0017	70±9.18, 0.0038	**74.62±5.83**, 0.0012
ecoli-0-1-4-6vs5 280,6	**97.5±2.71**, 0.006	**97.5±2.71**, 0.0161	96.07±2.65, 0.0034	96.07±3.43, 0.0018	73.33±5.58, 0.0037	74±4.35, 0.0015
ecoli-0-1-4-7vs2-3-5-6 336,7	96.73±0.67, 0.0078	96.73±0.67, 0.0262	93.75±1.93, 0.0037	94.93±3.28, 0.0022	96.43±2.82, 0.0051	**98.21±1.26**, 0.0015
ecoli-0-1-4-7vs5-6 332,6	**98.19±0.68**, 0.0078	**98.19±0.68**, 0.0187	95.18±1.95, 0.0029	95.17±2.72, 0.0019	95.04±2.37, 0.0037	96.12±3.27, 0.0013
ecoli-0-1vs2-3-5 244,7	96.73±3.09, 0.0045	96.73±3.09, 0.0304	94.68±3.7, 0.0034	94.66±4.94, 0.0025	95.49±1.84, 0.004	**97.59±1.35**, 0.0012
ecoli-0-1vs5 240,6	**97.92±2.55**, 0.0055	**97.92±2.55**, 0.0144	96.25±3.73, 0.0028	95.42±2.28, 0.0017	95.09±4.22, 0.0053	97.14±3.98, 0.0013
ecoli-0-2-3-4vs5 202,7	98.5±3.35, 0.004	98.5±3.35, 0.02	97.5±3.54, 0.0036	94.55±2.11, 0.003	95.42±4.52, 0.0037	**98.75±1.14**, 0.0011
ecoli-0-2-6-7vs3-5 224,7	96.44±5.79, 0.0042	96.44±5.79, 0.0112	91.99±6.39, 0.003	95.11±5.07, 0.0017	95.54±2.1, 0.0039	**98±3.26**, 0.0011
ecoli-0-3-4-6vs5 205,7	**97.07±3.18**, 0.004	**97.07±3.18**, 0.0151	95.61±4.01, 0.0031	94.63±2.04, 0.0017	93.28±6.52, 0.0048	96±5.53, 0.0011
ecoli-0-3-4-7vs5-6 257,7	**97.66±0.89**, 0.0052	**97.66±0.89**, 0.0129	94.95±2.6, 0.0037	94.53±2.58, 0.0029	95.12±3.45, 0.0048	97.56±2.44, 0.0011
ecoli-0-4-6vs5 203,6	97.05±2.67, 0.0037	97.05±2.67, 0.0206	95.57±3.61, 0.0019	94.07±3.31, 0.0017	94.17±4.09, 0.0056	**97.65±1.64**, 0.0011
ecoli-0-6-7vs3-5 222,7	96.4±3.02, 0.004	96.4±3.02, 0.0137	93.68±2.98, 0.0032	92.83±3.94, 0.0019	93.61±2.14, 0.0038	**98.01±2.09**, 0.0011
ecoli-0-6-7vs5 220,6	**97.27±1.9**, 0.0042	**97.27±1.9**, 0.011	93.64±6.31, 0.0028	94.55±4.13, 0.0015	94.19±5.34, 0.0038	96.87±2.99, 0.0011
ecoli0137vs26 311,7	96.15±3.3, 0.007	96.15±2.65, 0.0158	94.86±2.62, 0.0029	94.55±3.3, 0.0015	93.64±6.89, 0.0057	**96.82±2.59**, 0.0011
ecoli01vs5 240,7	98.33±1.74, 0.005	98.33±1.74, 0.015	**99.17±1.14**, 0.0038	98.75±1.86, 0.0015	95.19±3.39, 0.0039	97.12±2.35, 0.0011
ecoli3 336,7	92.85±3.87, 0.0084	92.85±3.87, 0.0267	93.75±3.23, 0.003	91.37±3.23, 0.0014	98.75±2.8, 0.0039	**99.58±0.93**, 0.0011
ecoli4 336,7	**98.52±1.48**, 0.0084	**98.52±1.48**, 0.0269	97.32±1.25, 0.0036	97.92±1.7, 0.0014	93.74±3.87, 0.004	94.94±3.75, 0.0011
glass2 214,9	92.05±2.12, 0.0036	92.05±2.12, 0.0092	92.05±2.12, 0.0029	92.05±2.12, 0.0015	97.32±1.94, 0.004	**99.11±0.81**, 0.0011
glass4 214,9	**97.19±3.05**, 0.005	**97.19±3.05**, 0.0139	96.25±2.7, 0.0034	96.27±2.64, 0.0015	92.49±4.6, 0.0036	93.91±3.58, 0.0011
glass5 214,9	96.73±2.07, 0.0053	96.73±2.07, 0.0184	95.79±1.95, 0.0029	96.74±2.65, 0.0015	96.28±3.89, 0.0008	**98.14±1.95**, 0.0011
haber 306,3	73.82±8.03, 0.0087	73.82±8.03, 0.0242	73.82±8.03, 0.0038	73.49±8.48, 0.0015	97.66±1.64, 0.0006	**98.14±1.95**, 0.0011
haberman 306,3	73.82±8.03, 0.0089	73.82±8.03, 0.0306	74.15±7.64, 0.0037	73.49±8.48, 0.0015	73.82±8.03, 0.0006	**74.8±7.29**, 0.0011
iono 351,33	80.91±5.76, 0.0079	80.91±5.76, 0.02	**81.51±10.99**, 0.0025	72.38±22.53, 0.0017	74.15±7.64, 0.0007	74.8±7.29, 0.0011
led7digit-0-2-4-5-6-7-8-9vs1 443,7	**96.17±0.99**, 0.0114	**96.17±0.99**, 0.0294	95.27±2.55, 0.0017	94.81±1.73, 0.0015	74.7±15.87, 0.0045	84.93±10.49, 0.0012
new-thyroid1 215,5	95.35±1.64, 0.0042	95.35±1.64, 0.0119	97.21±1.04, 0.0016	**97.67±2.33**, 0.0015	94.37±4.56, 0.0043	96.85±1.84, 0.0011
shuttle-6vs2-3 230,9	**100±0**, 0.005	**100±0**, 0.0176	**100±0**, 0.0017	**100±0**, 0.0015	98.14±1.95, 0.006	99.53±1.04, 0.0011

Table 3.2: The average classification accuracies, training times, and ranks of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on each 79 D1 category UCI and KEEL datasets.

Model	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
Dataset samples, features	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time	Acc. ± Std. , time
votes 435,16	89.89±4.97, 0.023	90.11±5.37, 0.0316	92.41±3.69, 0.0024	85.75±7.48, 0.0015	**100±0**, 0.0054	**100±0**, 0.0011
wpbc 194,33	76.29±10.16, 0.0038	76.29±10.16, 0.0135	76.29±10.16, 0.0027	76.29±10.16, 0.0015	86.9±5.67, 0.0052	**94.94±2.52**, 0.0011
yeast1vs7 459,8	**94.77±1.18**, 0.0158	**94.77±1.18**, 0.0507	93.47±2.54, 0.0026	93.47±2.54, 0.0015	76.29±10.16, 0.0062	79.38±3.63, 0.0011
yeast2vs8 483,8	**97.73±2.12**, 0.0205	**97.73±2.12**, 0.0457	97.31±2.15, 0.0027	97.11±2.77, 0.0015	93.47±2.54, 0.0054	94.77±1.41, 0.0011
bank 4521,16	88.67±0.49, 3.1119	89.03±0.42, 4.9583	88.48±0.55, 0.0043	88.48±0.55, 0.0029	96.06±5.13, 0.0008	**98.14±1.7**, 0.0011
blood 748,4	76.64±13.29, 0.0687	76.64±13.29, 0.117	76.24±14.98, 0.0034	76.24±14.98, 0.0028	88.48±0.55, 0.0247	**88.5±0.56**, 0.0016
breast cancer wisc diag 569,30	79.44±3.43, 0.0138	81.54±5.8, 0.0221	**84.27±4.69**, 0.0038	81.69±6.92, 0.0023	76.24±14.98, 0.0025	78.51±11.76, 0.0012
chess krvkp 3196,36	72.3±27.33, 2.0588	75.77±23.19, 3.2068	58.7±14.93, 0.0047	75.77±23.19, 0.0023	80.85±4.6, 0.0026	**87.42±6.62**, 0.0012
credit approval 690,15	**84.06±9.78**, 0.0299	**84.06±9.78**, 0.0397	77.25±6.85, 0.003	76.38±13.81, 0.0025	70.64±7.77, 0.0041	75.99±23.24, 0.0014
cylinder bands 512,35	60.87±17.95, 0.0118	61.07±17.6, 0.0169	65±8.83, 0.0034	64.79±14.01, 0.0028	76.23±8.33, 0.0026	**82.9±9.86**, 0.0012
ilpd indian liver 583,9	**71.35±5.09**, 0.0774	**71.35±5.09**, 0.0235	**71.35±5.09**, 0.0032	**71.35±5.09**, 0.0028	64.04±7.65, 0.0028	68.32±10.01, 0.0012
mammographic 961,5	**77.94±5.78**, 0.0917	**77.94±5.78**, 0.138	73.15±2.91, 0.0034	71.08±2.65, 0.003	72.74±5.76, 0.0042	77.21±2.02, 0.0013
oocytes trisopterus nucleus 2f 912,25	**67.99±6.85**, 0.0684	**67.99±6.85**, 0.099	64.26±7.03, 0.0033	59.55±10.69, 0.0026	62.38±8.58, 0.0025	66.55±7.25, 0.0014
pima 768,8	**70.58±2.36**, 0.0476	**70.58±2.36**, 0.0641	65.24±5.69, 0.0035	65.1±5.95, 0.0026	65.63±5.57, 0.0032	69.66±4.88, 0.0013
monk1 556,6	51.79±3.06, 0.0164	52.15±3.1, 0.0697	51.97±3.91, 0.0015	52.5±4.21, 0.0015	51.96±4.42, 0.0009	**53.04±5.14**, 0.0013
monk3 556,6	50.72±1.55, 0.0225	50.9±1.42, 0.0476	51.44±2.22, 0.0016	52.16±3.72, 0.0015	51.8±2.81, 0.0007	**53.06±5.3**, 0.0011
checkerboard data 690,14	82.17±2.44, 0.0656	**82.61±2.46**, 0.0944	76.67±2.53, 0.0033	73.62±4.98, 0.0028	78.26±5.05, 0.0043	81.01±1.07, 0.0014
statlog australian credit 690,14	67.97±1.65, 0.0416	67.97±1.65, 0.0592	67.97±1.57, 0.0031	**68.55±1.5**, 0.0025	67.97±1.94, 0.0033	68.41±1.41, 0.0013
transfusion 748,4	77.3±12.01, 0.0323	77.3±12.01, 0.0897	76.51±14.55, 0.0027	76.24±14.98, 0.0015	76.51±14.55, 0.0007	**78.64±10.99**, 0.0012
vowel 988,10	95.54±2.14, 0.0604	95.54±2.14, 0.1765	94.43±2.42, 0.0026	93.01±5.2, 0.0017	93.63±0.97, 0.0054	**95.95±3.23**, 0.0013
yeast-0-2-5-6vs3-7-8-9 1004,8	**93.22±2.26**, 0.0937	**93.22±2.26**, 0.2803	90.73±3.25, 0.0025	90.54±3.12, 0.0016	90.14±2.65, 0.0044	92.43±2.06, 0.0012
yeast-0-2-5-7-9vs3-6-8 1004,8	**96.22±0.9**, 0.0932	**96.22±0.9**, 0.2532	93.92±1.59, 0.0028	93.13±2.2, 0.0017	94.52±1.05, 0.0008	95.72±0.44, 0.0012
yeast-0-3-5-9vs7-8 506,8	91.7±2.68, 0.0197	91.7±2.68, 0.1017	91.3±3.66, 0.0021	90.71±2.86, 0.0015	90.12±2.53, 0.0049	**91.9±2.56**, 0.0011
yeast-0-5-6-7-9vs4 528,8	92.42±1.35, 0.0216	92.42±1.35, 0.0554	90.72±1.81, 0.0028	90.34±1.82, 0.0015	90.72±1.58, 0.007	**93.18±1.43**, 0.0012
titanic 2201,3	77.1±15.93, 0.4282	77.33±16.02, 0.5734	77.92±15.58, 0.0029	77.33±16.02, 0.0027	77.87±13.46, 0.0029	**79.05±15.04**, 0.0015
abalone9-18 731,7	95.36±3.31, 0.0326	95.36±3.31, 0.1077	94.4±4.5, 0.0042	94.95±4.5, 0.0025	95.22±3.76, 0.0052	**95.9±3.82**, 0.0014
aus 690,14	82.17±2.44, 0.0285	**82.61±2.46**, 0.0756	76.67±2.53, 0.0039	73.62±4.98, 0.0021	78.26±5.05, 0.0042	81.01±1.07, 0.0017
cmc 1473,9	69.54±18.95, 0.6173	79.99±22.08, 0.5047	81.62±20.33, 0.0032	**81.69±20.36**, 0.0017	81.62±20.33, 0.0043	81.62±20.33, 0.0014
ripley 1250,2	59.84±3.37, 0.27	59.84±3.37, 0.2029	59.84±3.37, 0.0032	59.84±3.37, 0.0017	59.84±3.37, 0.0008	**60.4±3.12**, 0.0013
yeast5 1484,8	97.57±2.03, 0.4438	97.57±2.03, 0.7824	97.03±2.45, 0.0036	97.17±2.41, 0.0017	97.03±2.45, 0.0042	**97.78±1.3**, 0.0014
ozone 2536,72	97.12±2.26, 0.3991	97.12±2.26, 0.4385	97.12±2.26, 0.0043	97.12±2.26, 0.0021	97.12±2.26, 0.0032	**97.2±2.11**, 0.002
spambase 4601,57	99.3±1.31, 96.8396	**99.39±1.36**, 1.6873	77.4±13.77, 0.0033	**99.39±1.36**, 0.0019	77.75±13.38, 0.0038	99.39±3.86, 0.0016
Avg. Acc. ± Avg. Std.	83.16±7.04	84.26±6.44	82.53±7.39	82.18±7.91	83.21±6.6	**86.35±5.06**
Avg. time	0.1304	0.1909	0.0031	0.0019	0.0037	**0.0012**
Avg. rank	3.35	2.96	3.96	4.45	4.12	**2.16**

Here, Avg., Acc. and Std. are acronyms used for average, accuracy, and standard deviation, respectively.
† signifies the proposed model while boldface and underline signify the best and second-best models, respectively.

The proposed $\mathcal{L}_{RoBoSS}$-SVM stands out with an average accuracy of 86.35% and a standard deviation of 5.06, the highest and most consistent performance among all the models. This suggests that $\mathcal{L}_{RoBoSS}$-SVM not only excels in overall accuracy but also maintains stable performance across diverse datasets, highlighting its reliability. In comparison, the baseline models $\mathcal{L}_{hinge}$-SVM, $\mathcal{L}_{pin}$-SVM, $\mathcal{L}_{LINEX}$-SVM, $\mathcal{L}_{qtse}$-SVM, and $\mathcal{L}_{wave}$-SVM show lower average accuracies of 83.16%, 84.26%, 82.53%, 82.18%, and 83.21%, respectively, with higher standard deviations, indicating more variability in their performance. In terms of training time, $\mathcal{L}_{RoBoSS}$-SVM exhibits the best average training time of 0.0012 seconds. While the baseline models have longer average training times, with $\mathcal{L}_{hinge}$-SVM at 0.1304, $\mathcal{L}_{pin}$-SVM at 0.1909, $\mathcal{L}_{LINEX}$-SVM at 0.0031, $\mathcal{L}_{qtse}$-SVM at 0.0019, and $\mathcal{L}_{wave}$-SVM at 0.0037 seconds. The average rank further underscores the superiority of $\mathcal{L}_{RoBoSS}$-SVM, with the lowest average rank of 2.16, indicating its consistent high performance across various datasets. In contrast, the baseline models have higher average ranks: $\mathcal{L}_{hinge}$-SVM at 3.35, $\mathcal{L}_{pin}$-SVM at 2.96, $\mathcal{L}_{LINEX}$-SVM at 3.96, $\mathcal{L}_{qtse}$-SVM at 4.45, and $\mathcal{L}_{wave}$-SVM at 4.12. These results collectively highlight the advantages of the proposed $\mathcal{L}_{RoBoSS}$-SVM model, showcasing its ability to deliver higher accuracy, faster training times, and more consistent performance compared to traditional SVM models. The bounded and sparse characteristics of the RoBoSS loss function help in mitigating the influence of outliers and ensuring that the model prioritizes the most critical samples. This leads to better generalization and efficiency, making $\mathcal{L}_{RoBoSS}$-SVM a robust and effective choice for various supervised learning tasks. The results for each of the 9 D2 category datasets are presented in Table 3.3. These results demonstrate the superior performance of the proposed $\mathcal{L}_{RoBoSS}$-SVM across several datasets. For instance, on the Musk2 dataset, $\mathcal{L}_{RoBoSS}$-SVM achieves an accuracy of 84.59% with a standard deviation of 34.46, which is identical to $\mathcal{L}_{pin}$-SVM, $\mathcal{L}_{LINEX}$-SVM, and $\mathcal{L}_{wave}$-SVM but with a significantly faster training time. On

the Ringnorm dataset, $\mathcal{L}_{RoBoSS}$-SVM achieves the highest accuracy of 52.22% with a standard deviation of 0.9, outperforming all other models. Similarly, on the Twonorm dataset, $\mathcal{L}_{RoBoSS}$-SVM attains the highest accuracy of 52.24% with a standard deviation of 2.01, again outperforming the competing models. The average accuracy of the $\mathcal{L}_{RoBoSS}$-SVM model across all nine datasets is 74.35%, which is higher compared to the baseline models: $\mathcal{L}_{hinge}$-SVM at 66.16%, $\mathcal{L}_{pin}$-SVM at 68.15%, $\mathcal{L}_{LINEX}$-SVM at 69.73%, $\mathcal{L}_{qtse}$-SVM at 73.2%, and $\mathcal{L}_{wave}$-SVM at 73.3%. The overall results convincingly demonstrate the superiority of the $\mathcal{L}_{RoBoSS}$-SVM model over traditional SVM models. These results validate the potential of the RoBoSS loss function in enhancing the robustness and efficiency of SVM models, making $\mathcal{L}_{RoBoSS}$-SVM a highly effective choice for complex and large-scale classification tasks.

Table 3.3: The classification accuracies and training times of the $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on 9 D2 category KEEL and UCI datasets.

Model	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
Dataset (samples, features)	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time
Musk2 (6598, 166)	80±44.72, 18.7863	**84.59±34.46**, 22.9542	**84.59±34.46**, 0.0048	81.02±17.85, 0.0023	**84.59±34.46**, 0.0035	**84.59±34.46**, 0.0029
Ringnorm (7400, 20)	50.5±1.29, 8.0734	50.95±0.88, 14.7853	51.15±0.62, 0.0033	51.03±1.03, 0.0019	51.19±0.54, 0.0027	**52.22±0.9**, 0.0018
Twonorm (7400, 20)	50.61±0.82, 5.0918	50.8±0.52, 28.8532	50.78±0.9, 0.0031	50.92±1.35, 0.0019	50.88±2.19, 0.0027	**52.24±2.01**, 0.0024
EEG Eye State (14980, 14)	55.12±25.92, 127.686	61.78±22.46, 192.8241	68.93±16.06, 0.0033	69.71±15.36, 0.002	70.35±15.6, 0.0039	**71.2±13.68**, 0.0017
Magic (19020,10)	82.84±9.8, 443.3522	82.88±9.72, 217.4496	65.3±25.25, 0.0043	**95.16±10.82**, 0.0021	90.38±17.47, 0.0032	95.16±33.91, 0.0023
Credit Default (30000, 23)	**77.89±1.56**, 247.5376	77.88±1.56, 1415.7059	77.88±1.56, 0.0062	77.88±1.56, 0.0034	77.88±1.56, 0.0229	77.88±1.56, 0.0069
Adult (48842, 14)	*	*	76.41±1.8, 0.0106	76.07±0.25, 0.0042	76.19±2.33, 0.0085	**77.94±1.51**, 0.0051
Connect4 (67557, 42)	*	*	75.38±3.78, 0.0118	75.38±3.78, 0.0057	75.38±3.78, 0.0103	**75.4±3.75**, 0.0082
Miniboone (130064, 50)	*	*	77.17±18.82, 0.0156	81.67±17.11, 0.0144	**82.85±6.98**, 0.0156	82.5±7.94, 0.0118
Avg Acc. ± Avg. Std.	66.16±14.02	68.15±11.6	69.73±11.47	73.2±7.68	73.3±9.43	**74.35±11.08**

Acc., Avg., and Std. stand for accuracy, average, and standard deviation, respectively. † signifies the proposed model.
* denote that MATLAB encounters an "out of memory" error.
Boldface highlights the top-performing model, while underlining indicates the second-best model.

3.5.3 Evaluation on Datasets with Introduced Outliers and Label Noise

To rigorously assess the robustness and generalization capabilities of the proposed $\mathcal{L}_{RoBoSS}$-SVM model, it is essential to evaluate its performance under challenging conditions. Real-world data often contains outliers and label noise, which can significantly impact the accuracy and reliability of machine learning models. Therefore, conducting an evaluation on datasets with artificially introduced outliers and label noise provides a comprehensive understanding of the model's resilience to such anomalies. In this study, we selected five diverse datasets, namely cylinder_bands, ionosphere, spectf, titanic, statlog_australian_credit. The methodology for introducing outliers and label noise into training dataset is discussed in the experimental setup described in this chapter.

Table 3.4 displays the classification accuracy of the $\mathcal{L}_{RoBoSS}$-SVM model alongside the compared models $\mathcal{L}_{hinge}$-SVM, $\mathcal{L}_{pin}$-SVM, $\mathcal{L}_{LINEX}$-SVM, $\mathcal{L}_{qtse}$-SVM, and $\mathcal{L}_{wave}$-SVM across datasets with 5%, 10%, 20%, and 30% outliers. The proposed $\mathcal{L}_{RoBoSS}$-SVM model consistently surpasses the compared models. Specifically, in four out of five datasets, the $\mathcal{L}_{RoBoSS}$-SVM model achieves the top position, while in one dataset, it secures the second-best position compared to the baseline models. This superior performance can be attributed to the bounded nature of the RoBoSS loss function, which mitigates the influence of extreme values, thereby maintaining high accuracy even when datasets are significantly contaminated with outliers. In terms of overall accuracy, the $\mathcal{L}_{RoBoSS}$-SVM model shows a total average accuracy of 76.67%, outperforming the baseline models $\mathcal{L}_{hinge}$-SVM (71.35%), $\mathcal{L}_{pin}$-SVM (72.75%), $\mathcal{L}_{LINEX}$-SVM (73.27%), $\mathcal{L}_{qtse}$-SVM (70.3%), and $\mathcal{L}_{wave}$-SVM (75.07%). This consistency across different levels of outliers underscores the $\mathcal{L}_{RoBoSS}$-SVM robustness and stability in handling outlier-prone data. In the context of evaluating the robustness of the RoBoSS-SVM in the presence of label noise, we have observed notable results, as

detailed in Table 3.5. Specifically, $\mathcal{L}_{wave}$-SVM achieves the best accuracy on two datasets and the second-best accuracy on three datasets. Similarly, $\mathcal{L}_{pin}$-SVM attains the best accuracy on three datasets and the second-highest result on two datasets. In terms of total average accuracy across all five datasets and noise ratios, $\mathcal{L}_{wave}$-SVM outperforms the baseline models with an average accuracy of 72.47%. Meanwhile, $\mathcal{L}_{RoBoSS}$-SVM achieves the second-best total average accuracy of 71.67%. These findings emphasize that $\mathcal{L}_{wave}$-SVM and $\mathcal{L}_{pin}$-SVM are particularly effective in environments with label noise, while the proposed $\mathcal{L}_{RoBoSS}$-SVM also demonstrates competitive performance. These outcomes align with existing literature, which suggests that loss functions incorporating penalties for correctly classified samples tend to be more effective in managing label noise [27, 67]. This insight underscores the potential for further enhancing the efficiency of the $\mathcal{L}_{RoBoSS}$-SVM by exploring modifications to the RoBoSS loss function. Future research could focus on redesigning the RoBoSS loss function to also penalize correctly classified samples to a certain extent while maintaining a balance with its sparsity property. This approach could improve its performance in scenarios with prevalent label noise.

Table 3.4: The classification accuracy of the proposed $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on datasets with varying levels of outliers.

Dataset	Outliers	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
cylinder_bands	5%	64.79	64.79	63.66	60.87	66.75	68.53
	10%	64.79	64.79	65.41	61.26	66.75	69.33
	20%	67.93	64.79	60.87	61.26	66.75	68.53
	30%	63.22	66.75	64.82	61.26	66.75	67.95
Avg.		65.19	65.28	63.69	61.17	66.75	**68.59**
ionosphere	5%	65.8	68.36	78.38	69.07	84.64	88.06
	10%	65.8	72.35	76.66	64.43	85.21	87.48
	20%	70.66	75.2	77.81	64.14	83.22	88.07
	30%	67.48	76.93	79.24	64.43	82.91	87.76
Avg.		67.43	73.21	78.02	65.52	83.99	**87.84**
spectf	5%	79.34	79.34	79.34	79.34	79.34	79.34
	10%	79.34	79.34	79.34	79.34	79.34	79.34
	20%	79.34	79.34	79.34	79.34	79.34	79.72
	30%	79.34	79.34	79.34	79.34	79.34	79.34
Avg.		79.34	79.34	79.34	79.34	79.34	**79.44**
titanic	5%	76.33	77.69	77.33	77.64	77.92	79.05
	10%	76.33	78.28	77.87	77.55	77.33	79.05
	20%	77.33	78.28	77.1	76.01	77.33	79.05
	30%	77.33	77.33	77.33	76.83	76.83	79.05
Avg.		76.83	77.89	77.41	77.01	77.35	**79.05**
statlog_australian_credit	5%	67.97	67.97	67.83	68.55	67.97	68.12
	10%	67.97	67.83	67.83	68.41	67.83	68.26
	20%	67.83	68.12	67.97	68.55	67.97	68.26
	30%	68.12	68.12	67.83	68.41	67.97	68.84
Avg.		67.97	68.01	67.86	**68.48**	67.93	68.37
Total Avg.		71.35	72.75	73.27	70.3	75.07	**76.67**

Here, Avg. denotes the average, and † signifies the proposed model.
Boldface highlights the top-performing model, while underlining indicates the second-best model.

Table 3.5: The classification accuracy of the proposed $\mathcal{L}_{RoBoSS}$-SVM along with the compared models on datasets with varying levels of label noise.

Dataset	Noise	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
cylinder_bands	5%	61.07	61.26	60.87	60.87	62.85	61.85
	10%	61.07	66.75	62.47	60.87	63.46	64.41
	20%	60.87	61.26	60.87	62.24	62.87	60.87
	30%	61.26	64.79	60.87	60.87	60.87	61.08
Avg.		61.07	**63.52**	61.27	61.22	62.51	62.05
ionosphere	5%	66.14	67.79	65.84	68.45	81.35	75.64
	10%	69.28	69.75	71.79	64.14	75.51	73.51
	20%	65.48	72.05	71.53	66.95	74.66	72.66
	30%	70.37	70.37	68.37	66.16	74.39	72.39
Avg.		67.82	69.99	69.38	66.42	**76.48**	73.55
spectf	5%	79.34	79.34	79.34	79.34	79.34	79.34
	10%	79.34	79.34	79.34	79.34	79.34	79.34
	20%	79.34	79.34	79.34	79.34	79.71	79.34
	30%	79.34	79.34	79.34	79.34	79.34	79.34
Avg.		79.34	79.34	79.34	79.34	**79.44**	79.34
titanic	5%	77.1	77.1	77.1	72.07	78.43	77.33
	10%	76.4	77.1	77.33	71.84	77.33	77.92
	20%	76.1	77.1	73.69	73.69	77.1	73.69
	30%	75.33	76.33	72.78	73.92	72.42	73.01
Avg.		76.23	**76.91**	75.22	72.88	76.32	75.49
statlog_australian_credit	5%	67.97	68.97	67.83	67.97	68.5	67.83
	10%	68.12	69.12	67.83	67.83	67.83	67.83
	20%	68.12	68.12	67.83	67.83	68.23	68.12
	30%	67.83	67.83	67.97	67.83	67.83	67.97
Avg.		68.01	**68.51**	67.86	67.86	68.1	67.93
Total Avg.		70.49	71.65	70.62	69.54	**72.47**	71.67

Here, Avg. denotes the average, and † signifies the proposed model.
Boldface highlights the top-performing model, while underlining indicates the second-best model.

3.5.4 Evaluation on Biomedical Datasets

In this subsection, we provide the experimental results on publicly available biomedical datasets. Specifically, the electroencephalogram (EEG) signal dataset and the breast cancer (BreaKHis) dataset.

The EEG data [81] includes five sets: A, B, O, C, and S. Each contains 100 single-channel EEG signals that were sampled at 173.61 hertz with a duration of 23.6 seconds. The sets O and C stand for the subject's eyes open and closed signals, respectively. Sets A and B provide the EEG signal that represents the subject's interictal state. The seizure activity signal is contained in set S. The feature selection process is the same as opted in [82]. Table 3.6 presents the dataset-wise EEG results. The results highlight the superior performance of the proposed $\mathcal{L}_{RoBoSS}$-SVM model. The $\mathcal{L}_{RoBoSS}$-SVM achieves an average accuracy of 79.42% with a standard deviation of 6.28, outperforming all baseline models. The closest competitor, $\mathcal{L}_{wave}$-SVM, has an average accuracy of 74.73% with a standard deviation of 6.98. The $\mathcal{L}_{RoBoSS}$-SVM model also demonstrates the lowest average training time of 0.0014 seconds and the best average rank of 1.06, indicating its overall efficiency and robustness.

Further, we evaluate the models on BreaKHis dataset [83]. We employed 1240 scans from the dataset at a magnification of 400 times. These scans are classified as either benign or malignant. The benign category includes four subclasses: phyllodes tumor (PT) with 115 scans, adenosis (AD) with 106 scans, fibroadenoma (FD) with 237 scans, and tubular adenoma (TA) with 130 scans. The malignant category is divided into lobular carcinoma (LC) with 137 scans, papillary carcinoma (PC) with 138 scans, ductal carcinoma (DC) with 208 scans, and mucinous carcinoma (MC) with 169 scans. To extract features, we employ the same process as outlined in [84]. Table 3.7 presents the dataset-wise BreaKHis results. The outcomes illustrate the dominance of the $\mathcal{L}_{RoBoSS}$-SVM model, achieving an average accu-

Table 3.6: The classification accuracies and training times of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on the EEG dataset.

Model	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
Dataset	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time
EEG_B_vs_S_bhattacharyya_100	69.5±5.97, 0.0049	70.5±8.55, 0.0051	73.5±7.83, 0.0015	55±1.77, 0.0017	74±10.4, 0.0025	**77±6.47**, 0.0016
EEG_B_vs_S_entropy_100	69.5±5.97, 0.0055	70.5±8.55, 0.004	73.5±7.83, 0.0018	55±1.77, 0.0018	74±10.4, 0.004	**77±6.47**, 0.0017
EEG_B_vs_S_roc_50	72.5±3.54, 0.0051	73.5±4.87, 0.003	74.5±5.7, 0.0016	58±14.3, 0.0019	72±5.42, 0.0028	**79±9.45**, 0.0015
EEG_B_vs_S_ttest_200	83.5±5.18, 0.0059	**83.5±4.18**, 0.0038	77±6.22, 0.0016	55±1.77, 0.0017	76±6.98, 0.0039	81±7.42, 0.0018
EEG_B_vs_S_wilcoxon_100	76±2.24, 0.0048	76±2.24, 0.0034	79.5±4.11, 0.0019	55±1.77, 0.0017	76.5±5.18, 0.004	**81.5±6.27**, 0.0013
EEG_B_vs_S_wilcoxon_200	81.5±8.4, 0.0052	81.5±8.4, 0.0035	80±7.07, 0.0018	55±1.77, 0.0019	79.5±8.18, 0.004	**82.5±5.86**, 0.0013
EEG_A_vs_S_bhattacharyya_100	71.5±4.18, 0.0057	71.5±4.18, 0.0038	68±4.47, 0.0018	56±1.37, 0.0018	71±6.02, 0.0039	**76±8.59**, 0.0013
EEG_A_vs_S_bhattacharyya_200	78±7.58, 0.0049	78±7.58, 0.004	70±8.29, 0.0016	55±1.77, 0.0018	70±8.29, 0.0039	**79.5±4.11**, 0.0013
EEG_A_vs_S_entropy_100	71.5±4.18, 0.0052	71.5±4.18, 0.0037	68±4.47, 0.0018	56±1.37, 0.0017	71±6.02, 0.004	**76±8.59**, 0.0012
EEG_A_vs_S_entropy_200	78±7.58, 0.0049	78±7.58, 0.004	70±8.29, 0.0019	55±1.77, 0.0018	70±8.29, 0.0026	**79.5±4.11**, 0.0013
EEG_A_vs_S_ttest_100	79±6.75, 0.0051	80±6.37, 0.0034	74.5±9.75, 0.0017	55±1.77, 0.0018	76±4.54, 0.0025	**81±5.76**, 0.0014
EEG_C_vs_B_roc_200	77±4.11, 0.005	77±4.11, 0.0036	77.5±7.29, 0.0017	55±1.77, 0.0018	82±4.47, 0.0025	**84.5±3.26**, 0.0013
EEG_C_vs_B_wilcoxon_200	84.5±4.11, 0.0047	86±5.18, 0.0031	86±8.4, 0.0015	55±1.77, 0.0017	88±6.47, 0.0026	**90±5**, 0.0013
EEG_C_vs_A_entropy_200	77±7.58, 0.0056	77±7.58, 0.004	74±10.55, 0.0016	55±1.77, 0.0017	76±5.18, 0.0026	**80±3.06**, 0.0012
EEG_C_vs_A_entropy_50	67.5±9.68, 0.0053	67.5±9.68, 0.0034	69±6.75, 0.0017	55±1.77, 0.0019	65.5±11.65, 0.0024	**72±12.67**, 0.0012
EEG_C_vs_A_roc_150	77±5.42, 0.0051	77.5±3.95, 0.0039	79.5±4.81, 0.0016	55±1.77, 0.0018	79.5±4.81, 0.0028	**85±7.29**, 0.0012
EEG_C_vs_A_roc_50	71.5±3.35, 0.0049	71.5±3.35, 0.0043	73.5±8.77, 0.0018	55±1.77, 0.0018	71.5±6.98, 0.0025	**77±5.42**, 0.0015
EEG_C_vs_A_ttest_100	76.5±7.83, 0.0051	76.5±7.83, 0.0036	77±4.81, 0.0016	55±1.77, 0.0021	76.5±7.2, 0.0025	**82±7.79**, 0.0012
EEG_C_vs_A_ttest_150	76.5±6.75, 0.0056	76.5±5.76, 0.0033	78.5±4.54, 0.0017	55±1.77, 0.0022	78.5±4.54, 0.0025	**84±2.85**, 0.0013
EEG_C_vs_A_ttest_200	79.5±5.97, 0.005	79.5±5.97, 0.0036	79±3.79, 0.0016	55±1.77, 0.0022	79±5.76, 0.0025	**85±5.59**, 0.0013
EEG_C_vs_A_ttest_50	74.5±7.79, 0.0051	74.5±7.79, 0.0039	75±9.01, 0.0017	55±9.84, 0.0021	76±4.54, 0.0024	**79.5±5.42**, 0.0022
EEG_C_vs_A_wilcoxon_50	78.5±5.18, 0.0049	78.5±4.87, 0.0032	79.5±6.94, 0.0018	56±11.81, 0.002	79.5±7.79, 0.0029	**84±6.98**, 0.0016
EEG_C_vs_S_bhattacharyya_100	60.5±6.94, 0.005	60.5±6.94, 0.0035	63.5±9.12, 0.0017	55±1.77, 0.0021	64.5±6.94, 0.0047	**72±11.1**, 0.0017
EEG_C_vs_S_bhattacharyya_150	66.5±7.83, 0.0051	66.5±7.83, 0.0033	68.5±9.78, 0.0016	55±1.77, 0.0026	69±9.12, 0.0028	**72±2.74**, 0.0012
EEG_C_vs_S_entropy_100	60.5±6.94, 0.0048	60.5±6.94, 0.0038	63.5±9.12, 0.0016	55±1.77, 0.0024	64.5±6.94, 0.0025	**72±11.1**, 0.0012
EEG_C_vs_S_entropy_150	66.5±7.83, 0.0052	66.5±7.83, 0.0037	68.5±9.78, 0.0017	55±1.77, 0.0024	69±9.12, 0.0025	**72±2.74**, 0.0013
EEG_C_vs_S_ttest_150	68±11.24, 0.0049	68±11.24, 0.0035	71±6.52, 0.0017	55±1.77, 0.0022	75.5±9.42, 0.0034	**76.5±7.62**, 0.0012
EEG_C_vs_S_ttest_200	69.5±8.18, 0.0047	69.5±8.18, 0.0036	74.5±5.97, 0.0015	55±1.77, 0.0021	74.5±5.97, 0.0038	**76.5±6.02**, 0.0012
EEG_C_vs_S_wilcoxon_100	64.5±8.91, 0.0048	64.5±7.79, 0.0031	68±10.37, 0.0017	55±1.77, 0.0021	67.5±10.31, 0.0025	**73.5±5.48**, 0.0013
EEG_O_vs_B_roc_150	82±2.74, 0.005	82±2.74, 0.0037	79±1.37, 0.0016	55±1.77, 0.0023	84±3.79, 0.0031	**86.5±3.79**, 0.0013
EEG_O_vs_B_ttest_50	71±4.87, 0.0051	73±6.22, 0.0029	77.5±7.91, 0.0019	55±1.77, 0.0021	77±8.91, 0.0025	**81±8.02**, 0.0014
EEG_O_vs_B_wilcoxon_150	82±2.74, 0.0047	82±2.74, 0.0037	79±1.37, 0.0017	55±1.77, 0.0022	84±3.79, 0.003	**86.5±3.79**, 0.0013
Avg Acc. ± Avg. Std.	73.8±6.17	74.05±6.29	74.06±6.91	55.19±2.7	74.73±6.98	**79.42±6.28**
Avg. time	0.0051	0.0036	0.0017	0.002	0.003	**0.0014**

Here, Avg., Acc. and Std. are acronyms used for average, accuracy, and standard deviation, respectively.
† signifies the proposed model while boldface and underline signify the best and second-best models, respectively.

racy of 63.25% with a standard deviation of 5.03. This performance surpasses that of all baseline models, with $\mathcal{L}_{wave}$-SVM being the closest at 60.32% accuracy and a standard deviation of 4.15. Other models like $\mathcal{L}_{hinge}$-SVM, $\mathcal{L}_{pin}$-SVM, $\mathcal{L}_{LINEX}$-SVM, and $\mathcal{L}_{qtse}$-SVM exhibit lower accuracies of 59.28%, 60.09%, 60.41%, and 59.6%, respectively. Moreover, $\mathcal{L}_{RoBoSS}$-SVM records an average training time of 0.0016 seconds and an average rank of 1.38, indicating its superior performance and efficiency in handling complex biomedical data.

Table 3.7: The classification accuracies and training times of the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models on the BreaKHis dataset.

Model	$\mathcal{L}_{hinge}$-SVM [6]	$\mathcal{L}_{pin}$-SVM [27]	$\mathcal{L}_{LINEX}$-SVM [31]	$\mathcal{L}_{qtse}$-SVM [69]	$\mathcal{L}_{wave}$-SVM [67]	$\mathcal{L}_{RoBoSS}$-SVM†
Dataset	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time	Acc. ± Std., time
ADvsDC	66.87±4.75, 0.005	66.87±4.75, 0.0169	66.24±4.16, 0.0029	66.24±4.16, 0.0011	66.24±4.16, 0.0055	**66.88±4.08**, 0.0021
ADvsLC	57.21±6.49, 0.0033	57.21±6.49, 0.0216	58.89±7.25, 0.0036	57.61±5.18, 0.0011	58.89±5.59, 0.0041	**62.59±5.98**, 0.0017
ADvsMC	61.82±7.27, 0.0045	62.18±6.97, 0.0147	61.45±6.35, 0.0038	61.45±6.35, 0.0011	61.45±6.35, 0.0063	**63.27±4.15**, 0.0017
ADvsPC	56.54±6.82, 0.0045	57.76±4.97, 0.0174	57.76±4.97, 0.0031	57.36±4.69, 0.0011	57.76±4.97, 0.0042	**61.91±5.68**, 0.0021
FDvsDC	53.26±2.04, 0.0104	53.26±2.04, 0.0318	56.63±2.33, 0.0035	53.26±2.04, 0.0013	56.18±2.1, 0.0054	**60±4.47**, 0.0018
FDvsLC	**66.84±3.76**, 0.0077	**66.84±3.76**, 0.0128	66.3±5.85, 0.0037	63.37±3.51, 0.0012	65.5±6.06, 0.0043	66.57±5.02, 0.0016
FDvsMC	58.62±2.23, 0.0082	58.62±2.23, 0.0553	60.6±3.73, 0.0036	58.37±2.21, 0.0012	59.61±3.91, 0.0042	**62.56±4.74**, 0.0017
FDvsPC	63.2±3.96, 0.0071	63.2±3.96, 0.0281	64±3.89, 0.0038	63.2±3.96, 0.0011	63.73±6.56, 0.0037	**66.93±5.45**, 0.0016
PTvsDC	64.73±5.35, 0.0053	64.73±5.35, 0.017	64.41±4.85, 0.0034	64.41±4.85, 0.0012	64.41±4.85, 0.0037	**66.26±2.45**, 0.0015
PTvsLC	55.11±9.65, 0.0046	56.72±6.05, 0.0105	59.11±4.6, 0.0034	58.31±4.94, 0.0011	59.13±4.05, 0.0075	**63.05±7.25**, 0.0014
PTvsMC	59.5±6.39, 0.0045	59.5±6.39, 0.0234	59.5±6.39, 0.0036	59.5±6.39, 0.0011	59.5±6.39, 0.0036	**62.31±4.44**, 0.0014
PTvsPC	54.97±8.45, 0.0047	57.32±4.51, 0.0837	57.32±4.51, 0.0033	57.71±4.75, 0.0011	57.32±4.51, 0.0044	**60.48±9.87**, 0.0013
TAvsDC	**64.21±6.99**, 0.0065	**64.21±6.99**, 0.0473	61.55±5.54, 0.0036	61.55±5.54, 0.0011	61.55±5.54, 0.0043	63.64±5.92, 0.0016
TAvsLC	51.28±8.39, 0.0045	56.56±4.32, 0.0374	59.94±7.93, 0.0033	56.56±4.32, 0.0012	61.06±6.47, 0.0038	**64.05±1.49**, 0.0014
TAvsMC	59.85±6.71, 0.004	**60.19±6.4**, 0.0167	56.51±6.01, 0.0035	56.51±6.01, 0.0011	56.51±6.01, 0.0051	59.2±5.95, 0.0017
TAvsPC	54.47±6.54, 0.0053	56.35±4.15, 0.0107	56.35±4.15, 0.0078	58.23±4.76, 0.0011	56.35±4.15, 0.01	**62.3±3.51**, 0.0018
Avg. Acc. ± Avg. Std.	59.28±5.99	60.09±4.96	60.41±5.16	59.6±4.6	60.32±4.15	**63.25±5.03**
Avg. time	0.0056	0.0278	0.0037	**0.0011**	0.005	0.0016

Here, Avg., Acc. and Std. are acronyms used for average, accuracy, and standard deviation, respectively.
† signifies the proposed model while boldface and underline signify the best and second-best models, respectively.

3.5.5 Statistical Analysis of RoBoSS-SVM Results

For statistical evaluation, we employed the Friedman test followed by the Nemenyi post hoc test to assess the relative performance of these models.

Friedman test: The Friedman test [24] is employed to statistically analyze the significance of the models. In this test, each model is ranked on each dataset separately, with the best-performing model securing rank 1, the second-best model getting rank 2, and so on. Under the null hypothesis, all the models are equivalent, i.e., the average rank of each model is equal. The Friedman statistic follows the chi-squared χ_F^2 distribution with $p-1$ degrees of

freedom (d.f.), where p denotes the number of models and is given by:

$$\chi_F^2 = \frac{12D}{p(p+1)} \left[\sum_e R_e^2 - \frac{p(p+1)^2}{4} \right], \tag{3.15}$$

where D denotes the number of datasets and R_e is the mean rank of eth of the p models. The Friedman statistic is undesirably conservative and thus a better statistic is derived by Iman and Davenport [62] as:

$$F_F = \frac{(D-1)\chi_F^2}{D(p-1) - \chi_F^2}, \tag{3.16}$$

which follows F distribution with $((p-1), (p-1)(D-1))$ d.f.. From the statistical F-distribution table, at 5% level of significance, we find the value of $F((p-1), (p-1)(D-1))$. If $F_F > F((p-1), (p-1)(D-1))$, we reject the null hypothesis. In this case, substantial differences exist among the models. Table 3.8 presents the results of the Friedman test on D1 category UCI and KEEL datasets, the EEG dataset, and the BreaKHis dataset. The outcomes demonstrate that significant differences exist among the proposed $\mathcal{L}_{RoBoSS}$-SVM and baseline models.

Nemenyi post hoc test: In Nemenyi post hoc test [25], all models are compared pairwise. The performance of the two models is substantially different if the corresponding mean ranks differ by a certain threshold value (critical difference, $C.D.$). If the difference between the average ranks of two models exceeds $C.D.$, the model with the lower mean rank is statistically better because rank 1 denotes the best performance. The $C.D.$ is calculated as:

$$C.D. = q_\alpha \sqrt{\frac{p(p+1)}{6D}}, \tag{3.17}$$

where q_α are based on the studentized range statistic divided by $\sqrt{2}$ and called critical value for the two-tailed Nemenyi test. At 5 % level of significance, we can simply calculate

that the values of $C.D.$ for D1 category UCI and KEEL datasets, the EEG dataset, and the BreaKHis dataset are 0.85, 1.33, and 1.88, respectively. Tables 3.9, 3.10, and 3.11 present the results of the Nemenyi post hoc test on D1 category UCI and KEEL datasets, the EEG dataset, and the BreaKHis dataset, respectively.

Table 3.8: Illustrate the results of the Friedman test on D1 category UCI and KEEL datasets, the EEG dataset, and the BreaKHis dataset.

Dataset	p	D	χ_F^2	F_F	$F((p-1),(p-1)(D-1))$	Significant difference (As per Friedman test)
D1 category dataset	6	79	81.442	20.26	2.24	Yes
EEG dataset	6	32	114.43	77.844	2.27	Yes
BreaKHis dataset	6	16	28.969	8.515	2.35	Yes

Table 3.9: Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on D1 category UCI and KEEL datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
$\mathcal{L}_{hinge}$-SVM [6]	3.35	1.19	Yes
$\mathcal{L}_{pin}$-SVM [27]	2.96	0.8	No
$\mathcal{L}_{LINEX}$-SVM [31]	3.96	1.8	Yes
$\mathcal{L}_{qtse}$-SVM [69]	4.45	2.29	Yes
$\mathcal{L}_{wave}$-SVM [67]	4.12	1.96	Yes
$\mathcal{L}_{RoBoSS}$-SVM (Proposed)	2.16	-	N/A

Table 3.10: Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on the EEG dataset.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
$\mathcal{L}_{hinge}$-SVM [6]	3.86	2.8	Yes
$\mathcal{L}_{pin}$-SVM [27]	3.63	2.57	Yes
$\mathcal{L}_{LINEX}$-SVM [31]	3.31	2.25	Yes
$\mathcal{L}_{qtse}$-SVM [69]	6	4.94	Yes
$\mathcal{L}_{wave}$-SVM [67]	3.14	2.08	Yes
$\mathcal{L}_{RoBoSS}$-SVM (Proposed)	1.06	-	N/A

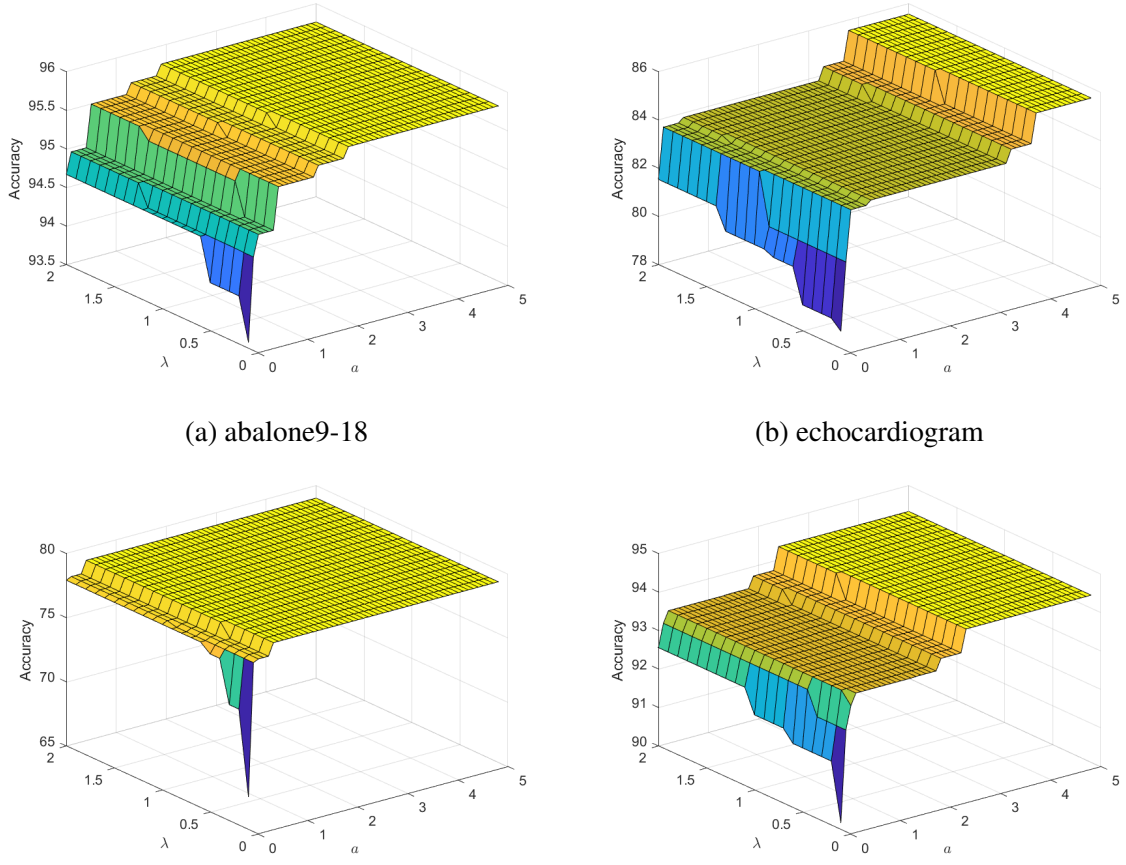
Table 3.11: Differences in the rankings of the proposed $\mathcal{L}_{RoBoSS}$-SVM model against baseline models on the BreaKHis dataset.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
$\mathcal{L}_{hinge}$-SVM [6]	4.22	2.84	Yes
$\mathcal{L}_{pin}$-SVM [27]	3.47	2.09	Yes
$\mathcal{L}_{LINEX}$-SVM [31]	3.72	2.34	Yes
$\mathcal{L}_{qtse}$-SVM [69]	4.59	3.21	Yes
$\mathcal{L}_{wave}$-SVM [67]	3.63	2.25	Yes
$\mathcal{L}_{RoBoSS}$-SVM (Proposed)	1.38	-	N/A

3.5.6 Sensitivity Analysis of RoBoSS-SVM

To understand the impact of the loss hyperparameters a and λ on the performance of the proposed $\mathcal{L}_{RoBoSS}$-SVM model, we conduct a sensitivity analysis using four diverse datasets: abalone9-18, echocardiogram, titanic, and ecoli3. The values of a and λ are varied systematically within the specified range while the other hyperparameters are fixed at their optimal values. For each combination of a and λ, the model's accuracy is recorded. The results are plotted in three-dimensional surface plots to visualize the sensitivity of the model's accuracy to changes in a and λ. The sensitivity plots for each of the four datasets are presented in Fig. 3.4. These plots highlight the intricate relationship between the hyperparameters a and λ, and the model's accuracy. Fig. 3.4a reveals that the accuracy of the abalone9–18 dataset stabilizes at higher values of a, with λ having a moderate influence. The model exhibits robustness across a wide range of λ values when a is sufficiently large. For the echocardiogram (see Fig. 3.4b), the accuracy shows a strong dependency on a, with higher values leading to improved performance. The impact of λ is less pronounced but still noticeable. The sensitivity plot for the titanic dataset (see Fig. 3.4c) indicates a consistent accuracy across various values of a and λ, with a notable dip in performance at lower values of both parameters. The model achieves its highest accuracy when both a and λ are set to higher values. For ecoli3 dataset (see Fig. 3.4d), the model's accuracy is highly sensitive to changes in both a and λ, with specific parameter combinations resulting in significantly higher accuracy. This analysis provides valuable insights into the behavior of the $\mathcal{L}_{RoBoSS}$-SVM model across different datasets. The key observations can be summarized as follows: (1) The parameter a plays a crucial role in determining the robustness and performance of the model. Higher values of a generally lead to improved accuracy, suggesting that the loss function's shape significantly impacts the model's ability to generalize. (2) The bounding parameter λ influences the model's performance, though its impact varies

across datasets. For some datasets, the choice of λ is critical, while for others, the model remains relatively stable across a wide range of λ values. (3) The interplay between a and λ is dataset-dependent, highlighting the need for dataset-specific tuning of these hyperparameters to achieve optimal performance. In conclusion, the sensitivity analysis underscores the importance of careful tuning of the loss hyperparameters a and λ to achieve optimal performance with the $\mathcal{L}_{RoBoSS}$-SVM model.



(a) abalone9-18

(b) echocardiogram

(c) titanic

(d) ecoli3

Figure 3.4: Sensitivity analysis of the $\mathcal{L}_{RoBoSS}$-SVM model with respect to the loss hyperparameters a and λ on abalone9-18, echocardiogram, titanic, and ecoli3 datasets. The 3D surface plots illustrate the model's accuracy variations as a and λ are systematically adjusted while keeping other hyperparameters fixed at their optimal values.

3.6 From RoBoSS-SVM to RoBoTS

The preceding sections established the RoBoSS loss as a robust and smooth surrogate and demonstrated its effectiveness within the SVM framework. The next step is to examine whether the same loss can improve twin support vector machines, where two non-parallel hyperplanes are learned through smaller optimization problems. This motivates the RoBoTS model and its inverse-free variant.

3.7 Twin SVM Background and Related Losses

This section only fixes the notation needed for the RoBoTS derivation. The twin-hyperplane principle and the main classification loss functions have already been developed in the foundational part of the thesis; therefore, the present chapter avoids repeating their complete background and focuses on how the RoBoSS loss is embedded into TSVM-type learning.

3.7.1 Notation and Modeling Link

Let the training set be $\{x_i, y_i\}_{i=1}^{m}$, where $x_i \in \mathbb{R}^{1\times n}$ and $y_i \in \{-1, 1\}$. Define the positive and negative data matrices as $X_+ \in \mathbb{R}^{m_+\times n}$ and $X_- \in \mathbb{R}^{m_-\times n}$, respectively, with $m = m_+ + m_-$. The vectors e_1 and e_2 denote all-one vectors of suitable dimensions.

RoBoTS follows the TSVM principle of learning two non-parallel hyperplanes, one associated with each class. The standard TSVM objective penalizes opposite-class deviations through hinge-type terms, while later TSVM variants replace this penalty by alternative losses such as pinball, LINEX, Wave, or Guardian losses. The present development uses the RoBoSS loss in this role so that the twin-hyperplane classifier inherits boundedness, smoothness, and sparsity. The next section gives the complete RoBoTS formulation, in-

cluding the inverse-free variant.

3.8 RoBoTS and Inverse-Free RoBoTS Models

3.8.1 Formulation and Optimization of RoBoTS Model

In this subsection, we provide the detailed formulation and optimization approach of the RoBoTS model for both linear and nonlinear cases. In the RoBoTS formulation, $\eta > 0$ is used as the local bounding parameter of the RoBoSS-type loss, playing the same role as λ in the RoBoSS-SVM part of this chapter. This convention is kept to avoid overloading λ, which is already widely used for regularization and other loss parameters in the thesis.

3.8.1.1 Linear Formulation

For a given training dataset, the objective of RoBoTS is to determine the positive and negative hyperplanes expressed as:

$$w_1^\top x + b_1 = 0 \quad \text{and} \quad w_2^\top x + b_2 = 0, \tag{3.18}$$

where $w_1, w_2 \in \mathbb{R}^{n\times 1}$ are the weight vectors and $b_1, b_2 \in \mathbb{R}$ represent the bias terms. To compute these hyperplanes, the linear RoBoTS framework is established by formulating the following optimization problem:

$$\begin{aligned} &\min_{w_1,b_1,\xi^-} \frac{1}{2}\sum_{i=1}^{m_+}\left(w_1^\top x_i + b_1\right)^2 + \frac{1}{2}C_1\left(\|w_1\|_2^2 + b_1^2\right) + C_2\sum_{j=1}^{m_-}\xi_j^- \\ &\text{s.t. } \xi_j^- = \eta\left[1 - \left(a\{1 + w_1^\top x_j + b_1\}_+ + 1\right)e^{\left(-a\{1+w_1^\top x_j+b_1\}_+\right)}\right], \quad j = 1,\ldots,m_-, \end{aligned} \tag{3.19}$$

$$\min_{w_2,b_2,\xi^+} \frac{1}{2}\sum_{j=1}^{m_-}\left(w_2^\top x_j + b_2\right)^2 + \frac{1}{2}C_3\left(\|w_2\|_2^2 + b_2^2\right) + C_4\sum_{i=1}^{m_+}\xi_i^+,$$
$$\text{s.t. } \xi_i^+ = \eta\left[1 - \left(a\{1 - w_2^\top x_i - b_2\}_+ + 1\right)e^{\left(-a\{1-w_2^\top x_i - b_2\}_+\right)}\right], \quad i = 1,\ldots,m_+, \quad (3.20)$$

where $\xi_+ = \left(\xi_1^+,\ldots,\xi_{m_+}^+\right)^\top \in \mathbb{R}^{m_+}, \xi_- = \left(\xi_1^-,\ldots,\xi_{m_-}^-\right)^\top \in \mathbb{R}^{m_-}$. For conciseness, we focus solely on the optimization problem (3.19), as the formulation for (3.20) is analogous. The objective function in Eqn. (3.19) comprises three distinct components, each contributing to the robustness and generalization of the proposed model:

1. The first component focuses on reducing the separation between the positive hyperplane and positive data points, ensuring tight alignment of the hyperplane with the positive class distribution.

2. The second component introduces an ℓ_2 regularization on the weight vector and bias, aligning with the principle of structural risk minimization to control model complexity and improve generalization performance.

3. The third component represents the aggregate penalty applied to all negative instances, calculated using the RoBoSS loss function. This term enhances the model's robustness by bounding the influence of outliers and promoting sparsity in the learned representation.

Due to the non-convexity of the RoBoSS loss function, the optimization problems (3.19) and (3.20) are inherently non-convex. Nevertheless, we leverage their smooth properties to devise an iterative algorithm for solving them. To facilitate computation, these problems are reformulated into a compact vector-matrix representation. The reformulated versions of

(3.19) and (3.20) are presented as follows:

$$\min_{\tilde{w_1}} P_1\left(\tilde{w_1}\right) = \frac{1}{2}\left\|G^\top \tilde{w_1}\right\|_2^2 + \frac{1}{2}C_1\left\|\tilde{w_1}\right\|_2^2 + C_2\mathcal{L}_1\left(\tilde{w_1}\right), \tag{3.21}$$

$$\min_{\tilde{w_2}} P_2\left(\tilde{w_2}\right) = \frac{1}{2}\left\|H^\top \tilde{w_2}\right\|_2^2 + \frac{1}{2}C_3\left\|\tilde{w_2}\right\|_2^2 + C_4\mathcal{L}_2\left(\tilde{w_2}\right). \tag{3.22}$$

The terms $\mathcal{L}_1\left(\tilde{w_1}\right)$ and $\mathcal{L}_2\left(\tilde{w_2}\right)$ are defined as follows:

$$\mathcal{L}_1\left(\tilde{w_1}\right) = \sum_{j=1}^{m_-} \eta\left[1 - \left(a\{1 + H_j^\top \tilde{w_1}\}_+ + 1\right)e^{\left(-a\{1+H_j^\top \tilde{w_1}\}_+\right)}\right],$$

$$\mathcal{L}_2\left(\tilde{w_2}\right) = \sum_{i=1}^{m_+} \eta\left[1 - \left(a\{1 - G_i^\top \tilde{w_2}\}_+ + 1\right)e^{\left(-a\{1-G_i^\top \tilde{w_2}\}_+\right)}\right].$$

Here, G_i and H_j denote the ith and j^{th} columns of matrices G and H, respectively. The matrices $G = [X_+, e_1]^\top \in \mathbb{R}^{(n+1)\times m_+}$ and $H = [X_-, e_2]^\top \in \mathbb{R}^{(n+1)\times m_-}$. Additionally, the weight vectors and biases are combined as $\tilde{w_1} = \left[w_1^\top, b_1\right]^\top \in \mathbb{R}^{n+1}$ and $\tilde{w_2} = \left[w_2^\top, b_2\right]^\top \in \mathbb{R}^{n+1}$. For simplicity, we denote $\{1+H_j^\top \tilde{w_1}\}_+$ and $\{1-G_i^\top \tilde{w_2}\}_+$ as A_j and B_i, respectively.

Now, based on the optimality conditions, we derive the following results:

$$\nabla P_1\left(\tilde{w_1}\right) = \left(GG^\top + C_1 I\right)\tilde{w_1} + \hat{H}s_1 = 0, \tag{3.23}$$

$$\nabla P_2\left(\tilde{w_2}\right) = \left(HH^\top + C_3 I\right)\tilde{w_2} - \hat{G}s_2 = 0. \tag{3.24}$$

Here, the terms $\hat{G}$ and $\hat{H}$ are defined as $\hat{G} = \left[C_4 G_1, \ldots, C_4 G_{m_+}\right] \in \mathbb{R}^{(n+1)\times m_+}$ and $\hat{H} = \left[C_2 H_1, \ldots, C_2 H_{m_-}\right] \in \mathbb{R}^{(n+1)\times m_-}$. The vectors s_1 and s_2 are expressed as $s_1 = \left[s_{11}, \ldots, s_{1m_-}\right]^\top \in \mathbb{R}^{m_-}$ and $s_2 = \left[s_{21}, \ldots, s_{2m_+}\right]^\top \in \mathbb{R}^{m_+}$, where each component is given by $s_{1j} = \eta a^2 A_j e^{-aA_j}$ and $s_{2i} = \eta a^2 B_i e^{-aB_i}$.

Now using Eqns. (3.23) and (3.24), we derive the iterative Eqns. for solving the opti-

mization problems (3.21) and (3.22) as follows:

$$\tilde{w_1}^{t+1} = -\left(GG^\top + C_1 I\right)^{-1} \hat{H} s_1^t, \tag{3.25}$$

$$\tilde{w_2}^{t+1} = \left(HH^\top + C_3 I\right)^{-1} \hat{G} s_2^t. \tag{3.26}$$

Here, t denotes the iteration index. The process iteratively applies Eqns. (3.25) and (3.26) until the algorithm converges. Once the solutions are obtained, the pair of hyperplanes defined in Eqn. (3.18) can be derived.

To classify an unseen sample $\widetilde{x} \in \mathbb{R}^{1\times n}$, the following decision rule is applied:

$$\text{Label of } \widetilde{x} = \begin{cases} +1, & \text{if } \frac{\left|w_1^\top \widetilde{x} + b_1\right|}{\|w_1\|} \leq \frac{\left|w_2^\top \widetilde{x} + b_2\right|}{\|w_2\|}, \\ -1, & \text{otherwise.} \end{cases} \tag{3.27}$$

3.8.1.2 Nonlinear Formulation

In nonlinear cases, data is mapped to a higher-dimensional feature space using the kernel trick, enabling the separation of data points in the transformed space. The goal of the nonlinear RoBoTS model is to find a pair of hypersurfaces, expressed as follows:

$$\mathcal{K}\left(x, X^\top\right) v_1 + b_1 = 0 \quad \text{and} \quad \mathcal{K}\left(x, X^\top\right) v_2 + b_2 = 0, \tag{3.28}$$

where $X = [X_+; X_-]^\top$ represents the combined dataset, and $\mathcal{K}(\cdot,\cdot)$ denotes the kernel function. To compute these hypersurfaces, the corresponding optimization problems are

formulated as follows:

$$\min_{v_1,b_1,\zeta^-} \sum_{i=1}^{m_+} \frac{1}{2}\left(\mathcal{K}\left(x_i, X^\top\right) v_1 + b_1\right)^2 + \frac{1}{2}C_1\left(\|v_1\|_2^2 + b_1^2\right) + C_2 \sum_{j=1}^{m_-} \zeta_j^-, \tag{3.29}$$

$$\text{s.t.} \quad \zeta_j^- = \eta\Bigg[1 - \Big(a\{1 + \mathcal{K}\left(x_j, X^\top\right) v_1 + b_1\}_+ + 1\Big) \times e^{\left(-a\{1+\mathcal{K}\left(x_j, X^\top\right)v_1+b_1\}_+\right)}\Bigg], \quad j = 1, \ldots, m_-,$$

$$\min_{v_2,b_2,\zeta^+} \sum_{j=1}^{m_-} \frac{1}{2}\left(\mathcal{K}\left(x_j, X^\top\right) v_2 + b_2\right)^2 + \frac{1}{2}C_3\left(\|v_2\|_2^2 + b_2^2\right) + C_4 \sum_{i=1}^{m_+} \zeta_i^+, \tag{3.30}$$

$$\text{s.t.} \quad \zeta_i^+ = \eta\Bigg[1 - \Big(a\{1 - \mathcal{K}\left(x_i, X^\top\right) v_2 - b_2\}_+ + 1\Big) \times e^{\left(-a\{1-\mathcal{K}\left(x_i, X^\top\right)v_2-b_2\}_+\right)}\Bigg], \quad i = 1, \ldots, m_+.$$

The vector-matrix forms of optimization problems (3.29) and (3.30) are presented as follows:

$$\min_{\tilde{v_1}} Q_1\left(\tilde{v_1}\right) = \frac{1}{2}\left\|R^\top \tilde{v_1}\right\|_2^2 + \frac{1}{2}C_1\left\|\tilde{v_1}\right\|_2^2 + C_2\mathcal{L}_1\left(\tilde{v_1}\right), \tag{3.31}$$

$$\min_{\tilde{v_2}} Q_2\left(\tilde{v_2}\right) = \frac{1}{2}\left\|S^\top \tilde{v_2}\right\|_2^2 + \frac{1}{2}C_3\left\|\tilde{v_2}\right\|_2^2 + C_4\mathcal{L}_2\left(\tilde{v_2}\right). \tag{3.32}$$

The terms $\mathcal{L}_1\left(\tilde{v_1}\right)$ and $\mathcal{L}_2\left(\tilde{v_2}\right)$ are defined as follows:

$$\mathcal{L}_1\left(\tilde{v_1}\right) = \sum_{j=1}^{m_-} \eta\Big[1 - \big(a\{1 + S_j^\top \tilde{v_1}\}_+ + 1\big)e^{\left(-a\{1+S_j^\top \tilde{v_1}\}_+\right)}\Big],$$

$$\mathcal{L}_2\left(\tilde{v_2}\right) = \sum_{i=1}^{m_+} \eta\Big[1 - \big(a\{1 - R_i^\top \tilde{v_2}\}_+ + 1\big)e^{\left(-a\{1-R_i^\top \tilde{v_2}\}_+\right)}\Big].$$

Here $R = \left[\mathcal{K}(X_+, X^\top), e_1\right]^\top \in \mathbb{R}^{(m+1)\times m_+}$ and $S = \left[\mathcal{K}(X_-, X^\top), e_2\right]^\top \in \mathbb{R}^{(m+1)\times m_-}$, where the columns of R and S are denoted as R_i and S_j, respectively. The weight vectors and bias terms are combined into $\tilde{v_1} = \left[v_1^\top, b_1\right]^\top$ and $\tilde{v_2} = \left[v_2^\top, b_2\right]^\top$.

Now, the gradients of Eqns. (3.31) and (3.32) are derived as follows:

$$\nabla Q_1\left(\tilde{v_1}\right) = \left(RR^\top + C_1 I\right)\tilde{v_1} + \hat{S}z_1 = 0, \tag{3.33}$$

$$\nabla Q_2\left(\tilde{v_2}\right) = \left(SS^\top + C_3 I\right)\tilde{v_2} - \hat{R}z_2 = 0. \tag{3.34}$$

Here, the terms $\hat{S}$ and $\hat{R}$ are defined as $\hat{S} = \left[C_2 S_1, \ldots, C_2 S_{m_-}\right] \in \mathbb{R}^{(m+1)\times m_-}$ and $\hat{R} = \left[C_4 R_1, \ldots, C_4 R_{m_+}\right] \in \mathbb{R}^{(m+1)\times m_+}$. Additionally, the vector $z_1 \in \mathbb{R}^{m_-}$ is composed of elements $z_{1j} = \eta a^2 E_j e^{-aE_j}$ for $j = 1, \ldots, m_-$, while $z_2 \in \mathbb{R}^{m_+}$ consists of elements $z_{2i} = \eta a^2 F_i e^{-aF_i}$ for $i = 1, \ldots, m_+$. E_j and F_i represent $\{1 + S_j^\top \tilde{v_1}\}_+$ and $\{1 - R_i^\top \tilde{v_2}\}_+$, respectively. The iterative expressions corresponding to Eqns. (3.33) and (3.34) are derived as follows:

$$\tilde{v_1}^{t+1} = -\left(RR^\top + C_1 I\right)^{-1} \cdot \left(\sum_{j=1}^{m_-} C_2 S_j \eta a^2 E_j e^{-aE_j}\right), \tag{3.35}$$

$$\tilde{v_2}^{t+1} = \left(SS^\top + C_3 I\right)^{-1} \cdot \left(\sum_{i=1}^{m_+} C_4 R_i \eta a^2 F_i e^{-aF_i}\right). \tag{3.36}$$

Here, t represents the iteration index. It is important to note that solving Eqns. (3.35) and (3.36) requires matrix inversion, which can become computationally intensive for large matrices. To handle this more efficiently, we utilize the Sherman-Morrison-Woodbury (SMW) theorem [85, 86], which reduces the dimensionality of the matrix involved in the inversion. With this approach, the inverse matrices in Eqns. (3.35) and (3.36) are substituted with the following expressions:

$$J_1 = \frac{1}{C_1}\left(I - R\left(C_1 I + R^\top R\right)^{-1} R^\top\right), \tag{3.37}$$

$$J_2 = \frac{1}{C_3}\left(I - S\left(C_3 I + S^\top S\right)^{-1} S^\top\right). \tag{3.38}$$

By using Eqns. (3.37) and (3.38), the iterative procedure can be written as follows:

$$\tilde{v_1}^{t+1} = -J_1 \hat{S} z_1^t, \tag{3.39}$$

$$\tilde{v_2}^{t+1} = J_2 \hat{R} z_2^t. \tag{3.40}$$

The iterative method involves alternately solving Eqns. (3.39) and (3.40) until the algorithm converges. Once the solutions v_1, b_1 and v_2, b_2 are obtained, the corresponding hypersurfaces generated by the kernel can be determined.

To determine the class label of a given sample $\widetilde{x} \in \mathbb{R}^{1\times n}$, the following decision rule is applied:

$$\text{Label of } \widetilde{x} = \begin{cases} +1, & \text{if } \frac{\left|\mathcal{K}\left(\widetilde{x}, X^\top\right) v_1 + b_1\right|}{\sqrt{v_1^\top \mathcal{K}\left(X, X^\top\right) v_1}} \leq \frac{\left|\mathcal{K}\left(\widetilde{x}, X^\top\right) v_2 + b_2\right|}{\sqrt{v_2^\top \mathcal{K}\left(X, X^\top\right) v_2}}, \\ -1, & \text{otherwise.} \end{cases} \tag{3.41}$$

The iterative algorithm for finding the parameters of the positive hyperplane of nonlinear RoBoTS is detailed in Algorithm 3.2, and the algorithm to find the negative hyperplane follows a similar structure.

Algorithm 3.2 Optimization Procedure for nonlinear RoBoTS

Input: Training data $\{(x_i, y_i)\}_{i=1}^{m}$, Gaussian kernel parameter σ, regularization parameters C_1, C_2; RoBoSS loss parameters a, η; convergence threshold θ; maximum iterations T.

Initialization: Set iteration counter $t = 0$, and initialize $\tilde{v}_1^{(0)} = 0$.

Compute:

$$R = \begin{bmatrix} \mathcal{K}(X_+, X) & e_1 \end{bmatrix}^\top, \quad S = \begin{bmatrix} \mathcal{K}(X_-, X) & e_2 \end{bmatrix}^\top$$

Construct:

$$J_1 = \frac{1}{C_1}\left(I - R(C_1 I + R^\top R)^{-1} R^\top\right),$$
$$\hat{S} = C_2 \cdot \begin{bmatrix} S_1 & S_2 & \dots & S_{m_-} \end{bmatrix}$$

while $t < T$ **do**

  Compute soft margins:

$$E_j^{(t)} = \max\left(0, 1 + S_j^\top \tilde{v}_1^{(t)}\right), \quad \forall j = 1, \dots, m_-$$

  Compute gradients:

$$z_{1j}^{(t)} = \eta a^2 E_j^{(t)} \cdot e^{-aE_j^{(t)}}, \quad z_1^{(t)} = [z_{11}^{(t)}, \dots, z_{1m_-}^{(t)}]^\top$$

  Update:

$$\tilde{v}_1^{(t+1)} = -J_1 \hat{S} z_1^{(t)}$$

  if $\|\tilde{v}_1^{(t+1)} - \tilde{v}_1^{(t)}\| < \theta$ **then**

    break

  else

    $t \leftarrow t + 1$

  end if

end while

Output: Extract final parameters v_1 and b_1 from:

$$\begin{bmatrix} v_1^\top, & b_1 \end{bmatrix}^\top = \tilde{v}_1^{(t+1)}$$

3.8.2 Inverse-Free RoBoTS (i-RoBoTS): An Alternate Optimization Approach

The RoBoTS model described in Section 3.8.1 is solved through an iterative optimization algorithm involving matrix inversion. Although this approach effectively addresses the optimization problem, it becomes computationally expensive when applied to large-scale datasets. The cubic complexity of matrix inversion renders it unsuitable for high-dimensional data or datasets with large sample sizes. To alleviate this computational bottleneck, we propose an alternative optimization strategy for RoBoTS using Nesterov's accelerated gradient (NAG)-based method. This technique preserves the original objective function but eliminates the need for matrix inversion, significantly enhancing the scalability of the model.

3.8.2.1 Linear Formulation

Unlike the inverse-based approach discussed in Section 3.8.1, here we directly minimize the objective functions $\mathcal{P}_1(\tilde{w}_1)$ and $\mathcal{P}_2(\tilde{w}_2)$, as defined in Eqns. (3.21) and (3.22), using their respective gradients derived in Eqns. (3.23) and (3.24). The optimization is performed iteratively using the NAG-based algorithm, which augments gradient descent with a lookahead step, thereby accelerating convergence. The NAG method is an improvement over the traditional gradient descent algorithm, incorporating a lookahead mechanism that improves convergence speed and stability. In contrast to standard momentum-based methods, NAG computes the gradient at a future (lookahead) position, providing more precise updates for faster convergence.

The NAG optimization algorithm comprises the following steps:

1. **Initialization:** Initialize $\tilde{w}_1^0 = 0$, $\tilde{w}_2^0 = 0$, $V_1^0 = 0$, and $V_2^0 = 0$. Select a learning rate α, momentum coefficient γ, stopping tolerance ϵ, and maximum iteration count T.

2. **Lookahead Step:**

$$\tilde{w}_1^{(t)} = \tilde{w}_1^{(t-1)} + \gamma V_1^{(t-1)},$$
$$\tilde{w}_2^{(t)} = \tilde{w}_2^{(t-1)} + \gamma V_2^{(t-1)}.$$

3. **Gradient Computation:** Use Eqns. (3.23) and (3.24) to evaluate the gradients:

$$g_1 = \nabla \mathcal{P}_1(\tilde{w}_1^{(t)}),$$
$$g_2 = \nabla \mathcal{P}_2(\tilde{w}_2^{(t)}).$$

4. **Velocity Update:**

$$V_1^{(t)} = \gamma V_1^{(t-1)} - \alpha g_1,$$
$$V_2^{(t)} = \gamma V_2^{(t-1)} - \alpha g_2.$$

5. **Parameter Update:**

$$\tilde{w}_1^{(t+1)} = \tilde{w}_1^{(t)} + V_1^{(t)},$$
$$\tilde{w}_2^{(t+1)} = \tilde{w}_2^{(t)} + V_2^{(t)}.$$

6. **Convergence Check:** If both conditions are satisfied:

$$\|\tilde{w}_1^{(t+1)} - \tilde{w}_1^{(t)}\| < \epsilon \text{ and } \|\tilde{w}_2^{(t+1)} - \tilde{w}_2^{(t)}\| < \epsilon,$$

or if the maximum number of iterations T is reached, then terminate the optimization process. Otherwise, proceed to the next iteration starting from Step 2.

3.8.2.2 Nonlinear Formulation

The NAG-based optimization strategy for the nonlinear RoBoTS model follows a similar procedure. Instead of optimizing over $\tilde{w}_1$ and $\tilde{w}_2$, we optimize over $\tilde{v}_1$ and $\tilde{v}_2$, as described in Eqns. (3.31) and (3.32), using the gradients defined in Eqns. (3.33) and (3.34). The optimization updates are identical to those described in the linear formulation. For implementation details, the procedure for estimating the parameters of the positive hyperplane is summarized in Algorithm 3.3. A similar methodology is employed for determining the negative hyperplane.

Algorithm 3.3 Optimization Procedure for nonlinear i-RoBoTS

1: **Input:** Training set $\{x_i, y_i\}_{i=1}^m$, Gaussian kernel parameter σ, regularization parameter C_1, C_2, RoBoSS loss parameters a, η, learning rate α, momentum coefficient γ, convergence threshold ϵ, maximum iteration count T.
2: Initialize $\tilde{v}_1^{(0)} = 0$, $V_1^{(0)} = 0$, $t = 0$.
3: **while** not converged and $t < T$ **do**
4: **Lookahead step:**

$$\tilde{v}_1^{(t)} = \tilde{v}_1^{(t-1)} + \gamma V_1^{(t-1)}$$

5: **Compute gradients:** Using Eqn. (3.33):

$$g_1 = \nabla \mathcal{Q}_1(\tilde{v}_1^{(t)})$$

6: **Update velocity:**

$$V_1^{(t)} = \gamma V_1^{(t-1)} - \alpha g_1$$

7: **Update parameters:**

$$\tilde{v}_1^{(t+1)} = \tilde{v}_1^{(t)} + V_1^{(t)}$$

8: **Convergence check:**
9: **if** $\|\tilde{v}_1^{(t+1)} - \tilde{v}_1^{(t)}\| < \epsilon$ **then**
10: **break**
11: **end if**
12: $t \leftarrow t + 1$
13: **end while**
14: **Output:** Extract final parameters v_1 and b_1 from:

$$\left[v_1^\top,\ b_1\right]^\top = \tilde{v}_1^{(t+1)}$$

3.8.3 Generalization Analysis of RoBoTS Model

In this subsection, we derive a generalization error bound for the proposed RoBoTS model. Our analysis leverages the key properties of the RoBoSS loss function, namely boundedness and Lipschitz continuity, and employs the framework of Rademacher complexity. The result provides a theoretical guarantee on the expected performance of the learned classifier under the RoBoTS framework.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ denote the training data drawn i.i.d. from an unknown distribution $\mathcal{P}$ over $\mathbb{R}^n \times \{-1, +1\}$. Let $\phi : \mathbb{R}^n \to \mathcal{H}$ denote the feature map induced by a Mercer kernel $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Consider the function class:

$$\mathcal{F}_B = \{ f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \ : \ \|\mathbf{w}\|_{\mathcal{H}} \leq B \} .$$

The goal is to minimize the expected risk under the RoBoSS loss:

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}} \left[\mathcal{L}_{\text{RoBoSS}}(1 - yf(\mathbf{x})) \right] ,$$

given only the empirical approximation:

$$\hat{R}_m(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{\text{RoBoSS}}(1 - y_i f(\mathbf{x}_i)) .$$

To bound $R(f)$ in terms of $\hat{R}_m(f)$, we use Rademacher complexity-based generalization theory [71], [87].

Theorem 3.8.1. *Let $\mathcal{L}_{RoBoSS} : \mathbb{R} \to [0, \eta]$ be a bounded loss function that is $\frac{\eta a}{e}$-Lipschitz. Then, for any $f \in \mathcal{F}_B$, with probability at least $1 - \delta$ over the draw of a training set of size*

m, the following generalization bound holds:

$$R(f) \leq \hat{R}_m(f) + \frac{2\eta a B \kappa}{e\sqrt{m}} + \eta\sqrt{\frac{\log(1/\delta)}{2m}},$$

where $\kappa^2 = \sup_{\mathbf{x}} \mathcal{K}(\mathbf{x}, \mathbf{x})$ denotes the bound on the kernel diagonal.

Proof. The proof proceeds in three steps.

Step 1: Rademacher generalization bound. Let $\mathfrak{R}_m(\mathcal{F}_B)$ denote the empirical Rademacher complexity of the function class $\mathcal{F}_B$:

$$\mathfrak{R}_m(\mathcal{F}_B) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}_B} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right],$$

where σ_i are i.i.d. Rademacher variables taking values in $\{-1, +1\}$ with equal probability.

For any loss function $\ell(z)$ that is L-Lipschitz and bounded by η, it is known [71] that with probability at least $1 - \delta$, the following holds:

$$R(f) \leq \hat{R}_m(f) + 2L \cdot \mathfrak{R}_m(\mathcal{F}_B) + \eta\sqrt{\frac{\log(1/\delta)}{2m}}.$$

Step 2: Lipschitz constant of RoBoSS loss. From Section III-B, we have shown that:

$$\left| \frac{d}{d\xi} \mathcal{L}_{\text{RoBoSS}}(\xi) \right| \leq \frac{\eta a}{e}, \quad \forall \xi \in \mathbb{R},$$

hence, the RoBoSS loss is $\frac{\eta a}{e}$-Lipschitz.

Step 3: Bounding the Rademacher complexity. For RKHS-induced functions of the form $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$, where $\|\mathbf{w}\|_{\mathcal{H}} \leq B$ and $\|\phi(\mathbf{x})\|_{\mathcal{H}}^2 = K(\mathbf{x}, \mathbf{x}) \leq \kappa^2$, the Rademacher complexity is bounded as (see [87]):

$$\mathfrak{R}_m(\mathcal{F}_B) \leq \frac{B\kappa}{\sqrt{m}}.$$

Putting it all together: Using $L = \frac{\eta a}{e}$ and the bound on $\mathfrak{R}_m(\mathcal{F}_B)$, we obtain:

$$R(f) \leq \hat{R}_m(f) + 2 \cdot \frac{\eta a}{e} \cdot \frac{B\kappa}{\sqrt{m}} + \eta \sqrt{\frac{\log(1/\delta)}{2m}}.$$

This concludes the proof. □

3.8.4 Computational Complexity

The computational complexity of RoBoTS is predominantly driven by the matrix inversion operations required during optimization. In the linear scenario, the model involves computing the inverse of two matrices, each of size $(n+1) \times (n+1)$, corresponding to the positive and negative class optimization problems. This results in a total complexity of $\mathcal{O}(2T \cdot (n+1)^3)$, where T denotes the maximum number of iterations. In the nonlinear case, the model computes the inverse of two kernel-induced matrices of dimensions $m_+ \times m_+$ and $m_- \times m_-$, yielding a computational complexity of $\mathcal{O}(T \cdot (m_+^3 + m_-^3))$. The cubic dependence on either the feature dimensionality (in the linear case) or the number of training samples per class (in the nonlinear case) makes the original RoBoTS formulation computationally demanding, especially for large-scale datasets. To address this limitation, the inverse-free RoBoTS variant employs NAG-based algorithm for optimization, thereby eliminating matrix inversion entirely. In the linear case, the computational cost per iteration is dominated by matrix-vector multiplications, leading to a total complexity of $\mathcal{O}(T \cdot m \cdot n)$. For the nonlinear formulation, where kernel evaluations dominate, the total complexity becomes $\mathcal{O}(T \cdot m^2)$. These improvements significantly enhance the scalability of RoBoTS across both linear and nonlinear settings.

3.9 RoBoTS Experimental Evaluation

This section aims to validate the effectiveness of the proposed RoBoTS models against baseline models. The evaluation is conducted using a suite of statistical metrics and tests, including accuracy, rank analysis, the Friedman test, the Nemenyi post-hoc test, and win-tie-loss comparisons, to ensure a thorough and reliable comparison. The evaluation is performed on several benchmark datasets from the KEEL and UCI repositories. Additionally, to demonstrate the applicability of the proposed RoBoTS model in real-world scenarios, we assess it on two biomedical datasets: the breast cancer dataset (BreaKHis) and the Alzheimer's dataset (ADNI).

3.9.1 Compared Models, Experimental Setup, and Hyperparameter Tuning

We evaluate the proposed RoBoTS models against a comprehensive set of benchmark algorithms. Specifically, we compare the linear RoBoTS model with SVM [6], TSVM [10], pinball loss-based generalized TSVM (Pin-GTSVM) [88], symmetric LINEX loss-based TSVM (SLTSVM) [89], wave loss-based TSVM (Wave-TSVM) [67], and RoBoSS loss-based SVM (RoBoSS-SVM) [90]. For the nonlinear RoBoTS model, we further broaden the comparison by incorporating representative ensemble and randomized neural network methods, specifically random forest (RF) [91], XGBoost (XGB) [92], random vector functional link network (RVFL) [16], and deep RVFL (dRVFL) [20], in addition to the kernel-based baselines used for the linear variant.

All experiments are implemented in MATLAB R2023a and executed on a Windows 10 workstation equipped with an Intel(R) Core(TM) i7-6700 processor (3.40 GHz, 4 cores, 8 threads) and 16 GB of RAM. To maintain consistency and ensure fair evaluation, all

datasets are preprocessed through feature normalization, transforming each attribute to have zero mean and unit variance. Model performance is assessed using a 4-fold cross-validation strategy, where in each fold, 75% of the samples are used for training and the remaining 25% for testing. For every hyperparameter configuration, training and testing are performed across all folds, and the highest testing accuracy obtained among them is reported as the final result for each dataset. This selection-and-reporting rule is applied identically to all RoBoTS baselines and proposed variants, which preserves fairness of the comparison under the adopted protocol. It does not, however, separate hyperparameter selection from final performance estimation. Selecting the best value from the testing folds can introduce optimistic selection bias [78]; hence, these results are conditional on the reported four-fold protocol and should be confirmed in future work through nested cross-validation or a separate validation and test design. The retained outputs likewise do not contain fold-level confusion matrices, so additional class-sensitive metrics cannot be reconstructed without rerunning the experiments.

For nonlinear models, we utilize the Gaussian kernel function. For SVM and RoBoSS-SVM the regularization parameter and the kernel parameter are selected from the set $\{10^i \,|\, i = -6, -4, \ldots, 4, 6\}$. Similarly, for TSVM-based models–including TSVM, Pin-GTSVM, SLTSVM, Wave-TSVM, and the proposed RoBoTS–the regularization parameters (C_2 and C_4), as well as the kernel parameter, are chosen from the same set. For Pin-GTSVM, the hyperparameter τ is selected from the values $\{0.1, 0.3, 0.5, 0.8, 1\}$. In the case of Wave-TSVM and RoBoSS-SVM, the loss function parameters are set to the same values as reported in [67] and [90], respectively. For the proposed RoBoTS models, the shape parameter (a) and the bounding parameter (η) are tuned within the ranges $[0.1 : 0.2 : 5.1]$ and $[0.5 : 0.5 : 1.5]$, respectively. In SLTSVM, Wave-TSVM, and RoBoTS, the structural risk minimization parameters (C_1 and C_3) are also selected from the

set $\{10^i \mid i = -6, -4, \ldots, 4, 6\}$. For these models, we set $C_1 = C_3$ and $C_2 = C_4$. To solve the optimization problems using the iterative algorithm in SLTSVM, Wave-TSVM, and proposed RoBoTS, the convergence precision (θ) is set to 10^{-6}, and the maximum number of iterations (T) is fixed at 50. For the i-RoBoTS model, the parameters of the NAG-based optimization algorithm are configured as follows: the learning rate α is set to 0.001, the momentum coefficient γ is 0.9, the convergence tolerance ϵ is 10^{-6}, and the maximum number of iterations T is set to 100. For random forest, the number of trees $\in \{50, 100, 200\}$, minimum leaf size $\in \{1, 3, 5, 10\}$, and maximum number of splits $\in \{10, 20, 50, 100\}$. For XGBoost, we explore maximum tree depth $\in \{3, 5, 7\}$, learning rate (η) $\in \{0.05, 0.1, 0.2\}$, number of boosting rounds $\in \{50, 100, 200\}$, subsample ratio $\in \{0.8, 1.0\}$, and column-sampling ratio $\in \{0.8, 1.0\}$. For the RVFL model, the ridge regularization parameter and and the number of hidden nodes are selected from $\{10^i \mid i = -6{:}2{:}6\}$ and $[3{:}20{:}203]$, respectively. For deep RVFL, hyperparameter tuning is performed in two stages: in Stage 1 (coarse search), the regularization parameter $\in \{10^i \mid i = -6{:}2{:}6\}$ and number of hidden nodes $\in \{256, 512, 1024\}$; in Stage 2 (refined search), the number of layers $\in \{1, 2, \ldots, 10\}$, and scaling as well as local perturbations of C and N are applied within $\pm 10\%$ of the best Stage-1 values to further optimize model performance. Both RVFL and deep RVFL employ the ReLU activation function in all experiments.

Table 3.12: Classification accuracy comparison of proposed linear RoBoTS with baseline models on UCI and KEEL datasets.

Dataset ↓ | Model →	SVM [6]	TSVM [10]	Pin-GTSVM [88]	SLTSVM [89]	Wave-TSVM [67]	RoBoSS-SVM [90]	i-RoBoTS†	RoBoTS†
breast_cancer (286, 9)	**100**	**100**	98.59	98.59	**100**	96.71	98.88	**100**
breast_cancer_wisc (699, 9)	95.43	96.68	**98.68**	78.16	91.43	75.59	76.99	93.10
congressional_voting (435, 16)	64.22	63.89	66.91	65.14	66.97	66.99	66.57	**67.89**
credit_approval (690, 15)	69.94	91.84	94.08	83.82	93.64	93.66	**98.91**	96.22
horse_colic (368, 25)	82.61	84.22	84.89	73.91	76.09	90.41	**90.55**	89.52
monks_1 (556, 6)	50.36	71.22	79.73	89.93	89.21	84.19	86.11	**93.53**
pima (768, 8)	75.52	76.21	75.40	71.88	73.96	80.59	**80.97**	75.48
pittsburg_bridges_T_OR_D (102, 7)	92.31	96.00	95.83	96.00	**100**	96.71	97.50	**100**
spect (265, 22)	71.64	72.27	73.67	68.18	77.27	79.13	**80.18**	76.24
statlog_heart (270, 13)	80.88	85.55	88.04	67.65	70.59	88.05	**89.30**	73.13
vertebral_column_2clases (310, 6)	**100**	**100**	**100**	**100**	**100**	96.71	99.24	**100**
aus (690, 14)	87.86	89.12	88.37	78.03	76.30	87.40	**89.81**	81.98
ecoli0137vs26 (311, 7)	93.59	92.12	93.13	92.31	94.81	95.47	**98.14**	97.40
ecoli01vs5 (240, 7)	**100**	93.24	93.25	95.00	95.00	91.88	94.03	95.00
ecoli3 (336, 7)	91.67	90.31	92.44	92.86	92.86	94.41	96.49	**97.62**
led7digit-0-2-4-5-6-7-8-9_vs_1 (443, 7)	93.69	96.40	94.32	93.69	**97.27**	94.97	96.91	96.40
monk1 (556, 6)	53.96	56.12	56.20	56.83	**60.43**	**60.43**	59.78	**60.43**
monk3 (554, 6)	51.80	54.68	55.95	56.83	58.27	58.44	59.50	**59.99**
shuttle-6_vs_2-3 (230, 9)	**100**	**100**	**100**	**100**	**100**	96.71	99.42	**100**
shuttle-c0-vs-c4 (1829, 9)	**100**	**100**	**100**	97.16	97.82	94.99	92.85	97.81
transfusion (748, 4)	59.36	87.17	83.87	87.17	87.70	86.82	87.04	**88.24**
votes (435, 16)	94.33	95.17	96.48	79.82	95.25	95.39	**98.02**	97.41
vowel (988, 10)	95.55	95.14	**96.43**	91.50	93.12	93.98	96.16	95.55
yeast-0-3-5-9_vs_7-8 (506, 8)	92.13	95.24	84.19	95.24	95.24	93.65	**95.80**	95.24
yeast1vs7 (483, 8)	94.78	92.10	92.78	94.78	94.78	95.66	94.02	**100**
Average Acc.	83.67	86.99	87.33	84.18	87.12	87.56	88.93	**89.13**
Average Rank	5.50	5.06	4.78	5.76	4.12	4.60	3.32	**2.86**
W, T, L of baseline w.r.t. RoBoTS	(6, 4, 15)	(5, 5, 15)	(5, 2, 18)	(0, 4, 21)	(3, 7, 15)	(5, 1, 19)	(11, 0, 14)	N/A

† denotes the proposed models. Best and second-best results are shown in bold and underlined, respectively.

(x, y, z) denotes the number of wins, ties, and losses of each baseline model when compared against the proposed RoBoTS model, respectively.

Table 3.13: Classification accuracy comparison of proposed nonlinear RoBoTS with baseline models on UCI and KEEL datasets.

Dataset ↓ | **Model** →	**SVM** [6]	**RF** [91]	**XGB** [92]	**RVFL** [16]	**TSVM** [10]	**Pin-GTSVM** [88]	**SLTSVM** [89]	**Wave-TSVM** [67]	**RoBoSS-SVM** [90]	**dRVFL** [20]	**i-RoBoTS**†	**RoBoTS**†
breast_cancer (286, 9)	**100**	56.22	54.84	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
breast_cancer_wisc (699, 9)	**100**	97.14	96.28	94.83	**100**	**100**	**100**	**100**	77.98	95.55	78.16	**100**
congressional_voting (435, 16)	65.14	62.07	62.07	64.14	63.30	67.62	66.97	66.97	**69.38**	65.74	67.89	67.89
credit_approval (690, 15)	84.39	86.38	85.51	95.27	98.84	90.37	97.11	96.00	97.50	98.42	**99.42**	**99.42**
horse_colic (368, 25)	76.09	88.86	86.14	87.13	88.04	86.76	85.87	83.00	92.20	89.30	**93.48**	89.13
monks_1 (556, 6)	72.66	73.20	76.62	90.24	73.38	87.74	78.42	81.00	86.00	**92.68**	87.05	86.33
pima (768, 8)	79.69	78.26	77.34	79.69	79.69	79.64	80.21	80.21	82.98	79.69	**83.33**	81.77
pittsburg_bridges_T_OR_D (102, 7)	96	89.08	88.12	96.15	96.15	97.92	96	**100**	**100**	**100**	**100**	**100**
spect (265, 22)	63.64	72.08	73.22	74.24	75.76	74.80	74.24	77.27	81.52	76.79	**81.82**	80.30
statlog_heart (270, 13)	74.63	85.22	83.00	83.58	85.07	85.31	88.06	88.06	89.80	85.55	**91.04**	89.71
vertebral_column_2clases (310, 6)	**100**	63.91	65.21	96.10	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
aus (690, 14)	84.97	88.41	88.26	91.91	88.95	87.93	90.70	91.28	89.79	91.91	**92.44**	91.86
ecoli0137vs26 (311, 7)	93.59	96.79	96.79	95.72	93.12	93.42	98.72	97.50	97.86	96.72	98.72	**100**
ecoli01vs5 (240, 7)	**100**	98.33	98.33	**100**	94.35	94.26	95.00	93.50	93.50	**100**	95.00	95.00
ecoli3 (336, 7)	91.67	94.05	93.16	95.62	91.40	93.40	**97.62**	**97.62**	96.80	96.62	**97.62**	**97.62**
led7digit-0-2-4-5-6-7-8-9_vs_1 (443, 7)	96.40	96.62	96.39	96.20	97.30	94.79	97.30	97.00	97.36	97.20	**98.20**	**98.20**
monk1 (556, 6)	53.96	51.26	51.80	56.83	55.40	57.34	55.40	55.40	**62.82**	56.83	61.15	56.83
monk3 (554, 6)	51.08	51.44	51.44	55.40	57.55	56.66	55.40	56.12	**60.83**	56.12	60.43	57.55
shuttle-6_vs_2-3 (230, 9)	**100**	**100**	98.71	**100**	**100**	**100**	**100**	**100**	99.10	**100**	**100**	**100**
shuttle-c0-vs-c4 (1829, 9)	**100**	**100**	99.89	**100**	**100**	**100**	**100**	**100**	97.38	**100**	94.09	**100**
transfusion (748, 4)	87.17	79.01	78.74	85.17	87.17	87.17	87.17	87.70	**88.70**	86.17	87.70	88.24
votes (435, 16)	89.91	96.33	96.56	96.08	98.17	98.57	99.07	99.07	97.78	97.08	99.07	**99.08**
vowel (988, 10)	95.14	97.67	97.87	95.38	99.19	**99.55**	95.95	98.38	95.90	96.60	97.17	98.79
yeast-0-3-5-9_vs_7-8 (506, 8)	93.65	91.70	91.31	93.24	95.24	84.11	95.24	95.50	96.04	95.83	**96.83**	95.24
yeast1vs7 (483, 8)	84.18	94.99	94.12	89.65	93.10	95.31	94.78	94.78	97.50	91.52	94.78	**100**
Average Acc.	85.36	83.56	83.27	88.50	88.45	88.51	89.17	89.45	89.95	89.85	90.22	**90.92**
Average Rank	9.04	8.74	9.84	7.70	6.80	6.68	6.02	5.48	4.88	5.62	3.72	**3.48**
W, T, L of baseline w.r.t. RoBoTS	(1, 5, 19)	(1, 2, 22)	(1, 0, 24)	(3, 4, 18)	(1, 7, 17)	(3, 5, 17)	(0, 8, 17)	(1, 7, 17)	(9, 3, 13)	(5, 6, 14)	(9, 9, 7)	N/A

† denotes the proposed model. Best and second-best results are shown in bold and underlined, respectively.

(x, y, z) denotes the number of wins, ties, and losses of each baseline model when compared against the proposed RoBoTS model, respectively.

3.9.2 Experimental Results on Benchmark KEEL and UCI Datasets

To evaluate the effectiveness of the proposed RoBoTS models, we downloaded 25 benchmark datasets obtained from the KEEL [79] and UCI [80] repositories. The selected datasets cover a broad range of domains, such as health and medicine (e.g., *breast_cancer*), business and finance (e.g., *credit_approval*), and biology (e.g., *ecoli3*), ensuring that the evaluation encompasses both homogeneous and heterogeneous feature distributions. The datasets vary in size and complexity, with the number of features ranging from 4 to 25 and the number of samples ranging from 102 to 1829, thereby enabling a comprehensive assessment of model generalization capability. The results for the linear models (Table 3.12) and nonlinear models (Table 3.13) clearly indicate that the proposed RoBoTS models consistently outperform their respective baseline across the majority of datasets.

For linear models (see Table 3.12), the proposed RoBoTS achieves the highest average accuracy of 89.13%, while the proposed i-RoBoTS model, optimized via an inverse-free strategy, delivers the second-best average accuracy of 88.93%, demonstrating that the integration of the RoBoSS loss function significantly enhances the robustness of linear TSVM classifiers. Notably, the RoBoSS-SVM model also performs strongly, achieving the third-highest average accuracy of 87.56%, which further affirms the general efficacy of the RoBoSS loss across different SVM variants. Other baseline models such as Pin-GTSVM and Wave-TSVM follow closely with average accuracies of 87.33% and 87.12%, respectively. Other models like TSVM (86.99%), SLTSVM (84.18%), and SVM (83.67%) rank lower, emphasizing the limitations of the unbounded loss functions. In terms of average rank, RoBoTS also achieves the best consistency with a rank of 2.86, better than i-RoBoTS at 3.32. Wave-TSVM secures the third-best average rank of 4.12, followed by RoBoSS-SVM and Pin-GTSVM with a rank of 4.60 and 4.78, respectively. TSVM achieves a rank of 5.06, while SLTSVM and SVM perform less consistently, with ranks of 5.76 and 5.50, re-

spectively. These rankings underscore the robustness and reliability of proposed models compared to baseline linear models.

For nonlinear models (see Table 3.13), the proposed RoBoTS model achieves the highest average classification accuracy of 90.92%, demonstrating its robustness and adaptability across diverse datasets. The inverse-free variant, i-RoBoTS, follows closely with an accuracy of 90.22%, confirming the consistency of the proposed framework. The RoBoSS-SVM ranks third with an accuracy of 89.95%, reaffirming the effectiveness of the RoBoSS loss function across different SVM-based architectures. Among the other competitive baselines, Wave-TSVM and SLTSVM achieve average accuracies of 89.45% and 89.17%, respectively, while Pin-GTSVM and TSVM record slightly lower performances of 88.51% and 88.45%. The RVFL and dRVFL models yield accuracies of 88.50% and 89.85%, indicating that although randomized neural networks perform well, the proposed RoBoTS framework provides superior stability and accuracy. Traditional ensemble methods such as random forest and XGBoost attain average accuracies of 83.56% and 83.27%, respectively, whereas the conventional SVM exhibits an accuracy of 85.36%. In terms of average rank, the proposed RoBoTS achieves the best ranking of 3.48, followed by i-RoBoTS with 3.72. RoBoSS-SVM, Wave-TSVM, and SLTSVM follow with ranks of 4.88, 5.48, and 6.02, respectively. Classical methods such as TSVM (6.80), Pin-GTSVM (6.68), RVFL (7.70), and SVM (9.04) show comparatively higher average ranks, indicating less consistent performance across datasets. These results collectively highlight the superiority and generalization capability of the proposed RoBoTS models over both kernel-based and ensemble-based baselines.

To further substantiate the superior performance of the proposed models compared to baseline methods, we conduct several statistical tests, including the Friedman test, the Nemenyi post-hoc test, and win-tie-loss comparisons to provide a comprehensive evaluation of the results.

Friedman test: The Friedman test [25] is conducted to determine whether the performance differences among the models are statistically significant. This test evaluates the relative performance of models by ranking them across multiple datasets. For each dataset, the model with the highest performance is given a rank of 1, followed by the second-best model with a rank of 2, and so on. According to the null hypothesis, all models are considered equivalent, implying that their average ranks do not significantly differ. The findings of the Friedman test for both linear and nonlinear models are summarized in Table 3.14. For linear models, with $l = 8$ and $\mathcal{D} = 25$, the computed values are $\chi_F^2 = 30.0667$ and $F_F = 4.9788$. For nonlinear models, with $l = 12$ and $\mathcal{D} = 25$, the corresponding values are $\chi_F^2 = 87.8954$ and $F_F = 11.2744$. In both cases, the computed F_F statistics exceed the critical values, leading to the rejection of the null hypothesis that all models perform equivalently. This result confirms that the observed performance differences among the models are statistically significant.

Table 3.14: Friedman test results for linear and nonlinear RoBoTS models against baseline models on KEEL and UCI datasets.

Model type	l	$\mathcal{D}$	χ_F^2	F_F	$F((l-1),(l-1)(\mathcal{D}-1))$	Significant difference (As per Friedman test)
Linear	8	25	30.0667	4.9788	2.06	Yes
nonlinear	12	25	87.8954	11.2744	1.82	Yes

Table 3.15: Nemenyi post-hoc test results for the linear RoBoTS model against baseline models on KEEL and UCI datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post-hoc test)
SVM [6]	5.5	2.64	Yes
TSVM [10]	5.06	2.2	Yes
Pin-GTSVM [88]	4.78	1.92	Yes
SLTSVM [89]	5.76	2.9	Yes
Wave-TSVM [67]	4.12	1.26	No
RoBoSS-SVM [90]	4.6	1.74	No
RoBoTS†	2.86	-	N/A

† denotes the proposed model.

Nemenyi post-hoc test: The Nemenyi post-hoc test [25] is employed to identify pairwise differences between models after determining that performance differences exist. This test assesses whether the mean ranks of two models vary beyond a predefined threshold known

Table 3.16: Nemenyi post-hoc test results for the nonlinear RoBoTS model against baseline models on KEEL and UCI datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post-hoc test)
SVM [6]	9.04	5.56	Yes
RF [91]	8.74	5.26	Yes
XGB [92]	9.84	6.36	Yes
RVFL [16]	7.70	4.22	Yes
TSVM [10]	6.80	3.32	Yes
Pin-GTSVM [88]	6.68	3.20	Yes
SLTSVM [89]	6.02	2.54	No
Wave-TSVM [67]	5.48	2.00	No
RoBoSS-SVM [90]	4.88	1.40	No
dRVFL [20]	5.62	2.14	No
RoBoTS†	3.48	-	N/A

† denotes the proposed model.

as the critical difference ($C.D.$). If the deviation between the mean ranks of two models surpasses the $C.D.$, the model with the lower mean rank is considered statistically better because rank 1 denotes the best performance. In this study, at a 10% significance level, the computed value of $C.D.$ is 1.92 for the linear models and 3.09 for the nonlinear models. Tables 3.15 and 3.16 provide the results of the Nemenyi post-hoc test for linear and nonlinear models, respectively. For linear models (see Table 3.15), the following observations are made: RoBoTS exhibits a rank difference of 2.9 from SLTSVM, 2.64 from SVM, 2.2 from TSVM, and 1.92 from Pin-GTSVM. These differences exceed the critical value and confirm that the RoBoTS model significantly outperforms these classical formulations. In contrast, the rank differences between RoBoTS and other baselines are below the critical threshold and thus not statistically significant. Specifically, the rank difference with RoBoSS-SVM is 1.74, and with Wave-TSVM is 1.26. Although these differences suggest that RoBoTS performs better in terms of rank. Despite some rank differences not reaching statistical significance, the combined results from accuracy, rank, and the Friedman test clearly highlight the remarkable performance of RoBoTS compared to all baseline models. For nonlinear models (see Table 3.16), the proposed RoBoTS model achieves the best average rank of 3.48. The rank differences between RoBoTS and several baseline models, SVM (5.56), RF (5.26), XGB (6.36), RVFL (4.22), TSVM (3.32), and Pin-GTSVM (3.20), all exceed

the critical difference value of 3.09, indicating that the proposed RoBoTS model performs significantly better than these methods. In contrast, the rank differences with SLTSVM (2.54), Wave-TSVM (2.00), RoBoSS-SVM (1.40), and dRVFL (2.14) fall below the critical difference threshold, suggesting that while RoBoTS maintains the best overall rank, its improvements over these models are not statistically significant. Nonetheless, when considered alongside the average accuracy and Friedman test results, these outcomes substantiate the robustness and consistent superiority of the proposed RoBoTS model against baseline models.

Win-tie-loss test: The win-tie-loss test [25] is a commonly used pairwise statistical method to determine whether there is a significant difference between the performances of two models. Under the null hypothesis, it is assumed that the two models are equivalent, meaning that each model is expected to win on $\mathcal{D}/2$ out of $\mathcal{D}$ datasets, where $\mathcal{D}$ is the total number of datasets. At a 5% level of significance, two models are considered significantly different if one model achieves at least $\mathcal{D}/2 + 1.96\sqrt{\mathcal{D}}/2$ wins. In cases where there is a tie, the score is shared equally between the two models. For $\mathcal{D} = 25$, the threshold for a significant difference is calculated as $\mathcal{D}/2 + 1.96\sqrt{\mathcal{D}}/2 = 17.4$ wins. The win-tie-loss analysis for linear and nonlinear models is presented at the bottom of Tables 3.12 and 3.13, respectively. For clarity, the triplet notation (x, y, z) represents the number of wins, ties, and losses of each baseline model when compared against the proposed RoBoTS model, respectively. For the linear models, RoBoTS demonstrates clear superiority over the majority of baselines. Specifically, when compared with SVM (6, 4, 15), TSVM (5, 5, 15), Pin-GTSVM (5, 2, 18), SLTSVM (0, 4, 21), Wave-TSVM (3, 7, 15), and RoBoSS-SVM (5, 1, 19), the number of losses for each baseline far exceeds the wins, indicating that RoBoTS outperforms all competing methods on most datasets. Even when compared with its inverse-free counterpart, i-RoBoTS (11, 0, 14), the results show competitive performance, with both variants

performing strongly across datasets. For the nonlinear models, RoBoTS again exhibits dominant performance. Baseline methods such as SVM (1, 5, 19), RF (1, 2, 22), XGB (1, 0, 24), RVFL (3, 4, 18), TSVM (1, 7, 17), and Pin-GTSVM (3, 5, 17) record significantly fewer wins than losses, confirming that RoBoTS outperforms them across most datasets. Although SLTSVM (0, 8, 17), Wave-TSVM (1, 7, 17), RoBoSS-SVM (9, 3, 13), and dRVFL (5, 6, 14) achieve closer competition, their win counts remain well below the 17.4 threshold required for statistical significance. The i-RoBoTS model (9, 9, 7) performs comparably to RoBoTS, further validating the stability and reliability of the proposed framework. Overall, after distributing ties equally between the compared models, RoBoTS consistently outperforms the majority of baselines in both linear and nonlinear settings.

These findings, together with the Friedman and Nemenyi test outcomes, provide strong evidence of the robustness and general effectiveness of the proposed RoBoTS models across diverse datasets and configurations.

Table 3.17: Classification accuracy comparison of proposed RoBoTS with baseline models on the BreaKHis dataset.

Dataset ↓ | Model →	**SVM** [6]	**RF** [91]	**XGB** [92]	**RVFL** [16]	**TSVM** [10]	**Pin-GTSVM** [88]	**SLTSVM** [89]	**Wave-TSVM** [67]	**RoBoSS-SVM** [90]	**dRVFL** [20]	**RoBoTS**†
ad_vs_dc (314, 768)	70.89	84.42	84.72	70.89	88.46	**91.46**	88.46	89.74	90.21	74.68	91.03
ad_vs_lc (243, 768)	63.93	60.48	61.29	73.33	72.13	69.09	63.93	68.85	68.41	70.00	**77.05**
ad_vs_mc (275, 768)	66.67	67.62	64.71	70.59	72.46	70.88	68.12	**73.91**	73.06	72.46	**73.91**
ad_vs_pc (244, 768)	63.93	62.30	62.30	68.85	67.21	**77.15**	67.21	70.49	70.90	67.21	72.13
fd_vs_dc (445, 768)	64.86	83.15	82.02	69.37	86.49	78.84	84.68	86.49	86.25	71.17	**89.19**
fd_vs_lc (374, 768)	66.67	68.21	66.86	68.82	**70.97**	70.44	**70.97**	69.89	69.42	**70.97**	**70.97**
fd_vs_mc (406, 768)	64.36	62.57	60.85	65.69	65.35	63.02	64.36	66.67	67.15	67.65	**68.32**
fd_vs_pc (375, 768)	66.67	68.55	68.81	72.34	69.15	68.13	67.74	69.15	69.64	73.12	**75.27**
pt_vs_dc (323, 768)	67.90	86.69	85.15	70.37	**91.36**	90.33	88.75	90.12	89.55	72.84	**91.36**
pt_vs_lc (252, 768)	58.73	57.54	57.14	60.32	61.90	60.82	65.08	**66.67**	66.20	**66.67**	**66.67**
pt_vs_mc (284, 768)	63.38	65.85	63.73	66.20	74.65	**74.68**	71.83	71.83	71.29	66.20	71.83
pt_vs_pc (253, 768)	59.38	64.83	63.20	62.50	68.25	**76.36**	70.31	70.31	70.68	65.08	70.31
ta_vs_dc (338, 768)	81.18	77.79	79.25	69.05	82.35	80.24	81.18	82.35	81.79	72.94	**84.71**
ta_vs_lc (267, 768)	61.19	73.40	72.27	65.67	**77.61**	76.59	74.63	72.73	72.32	71.21	74.24
ta_vs_mc (299, 768)	60.00	68.86	67.55	65.33	70.67	70.56	65.33	69.33	68.88	64.00	**72.00**
ta_vs_pc (268, 768)	62.69	66.42	58.96	61.19	67.16	68.78	67.16	68.66	69.11	68.66	**74.63**
Average Acc.	65.15	69.92	68.68	67.53	74.14	74.21	72.48	74.20	74.05	69.68	**76.48**
Average Rank	9.81	8.34	9.09	7.84	4.00	4.44	6.03	4.03	4.31	6.19	**1.91**

† denotes the proposed model. Best and second-best results are in bold and underlined, respectively.

Table 3.18: Classification accuracy comparison of proposed RoBoTS with baseline models on the ADNI dataset.

Dataset ↓ | Model →	**SVM** [6]	**RF** [91]	**XGB** [92]	**RVFL** [16]	**TSVM** [10]	**Pin-GTSVM** [88]	**SLTSVM** [89]	**Wave-TSVM** [67]	**RoBoSS-SVM** [90]	**dRVFL** [20]	**RoBoTS**†
CN_VS_AD (415, 91)	76.92	87.71	87.47	91.23	91.35	91.59	91.35	<u>93.27</u>	92.37	93.15	**94.23**
CN_VS_MCI (626, 91)	75.16	72.35	72.99	76.07	<u>77.07</u>	76.41	72.61	75.16	74.26	76.43	**78.98**
MCI_VS_AD (585, 91)	72.60	72.65	71.79	77.51	76.71	75.07	78.77	78.77	77.87	<u>79.19</u>	**80.82**
Average Acc.	74.89	77.57	77.42	81.60	81.71	81.02	80.91	82.40	81.50	<u>82.92</u>	**84.68**
Average Rank	9.17	9.67	10.00	6.33	5.17	5.67	6.67	4.00	5.67	<u>2.67</u>	**1.00**

† denotes the proposed model. Best and second-best results are in bold and underlined, respectively.

3.9.3 Experimental Results on Biomedical Datasets

To demonstrate the practical applicability and effectiveness of the proposed RoBoTS model in real-world scenarios, we evaluate its performance using the proposed nonlinear RoBoTS model on two widely used biomedical datasets: the BreaKHis dataset, a breast cancer dataset, and the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset.

- **BreaKHis dataset:** The BreaKHis dataset [83] comprises histopathological images, and for this study, we utilize 1,240 scans captured at a magnification of 400 times. These scans are categorized into benign and malignant classes, each further subdivided into specific subclasses. The benign class includes phyllodes tumor (PT) with 115 scans, adenosis (AN) with 106 scans, fibroadenoma (FD) with 237 scans, and tubular adenoma (TA) with 130 scans. The malignant class consists of lobular carcinoma (LC) with 137 scans, papillary carcinoma (PC) with 138 scans, ductal carcinoma (DC) with 208 scans, and mucinous carcinoma (MC) with 169 scans. Feature extraction is performed following the same methodology as described in [84]. Table 3.17 presents the performance comparison of the proposed nonlinear RoBoTS model against the baseline models on the BreaKHis dataset, highlighting the superior classification capability of the proposed RoBoTS model. RoBoTS achieves the highest average accuracy of 76.48%, outperforming all baseline models. Among the baseline methods, TSVM, Wave-TSVM, and Pin-GTSVM show competitive results

with average accuracies of 74.14%, 74.20%, and 74.21%, respectively. SLTSVM and RoBoSS-SVM record accuracies of 72.48% and 74.05%, while ensemble and randomized neural network models such as RF, XGB, RVFL, and dRVFL attain lower accuracies of 69.92%, 68.68%, 67.53%, and 69.68%, respectively. SVM performs the least effectively, with an average accuracy of 65.15%. The difference in accuracy between RoBoTS and the best-performing baseline model, Pin-GTSVM, is 2.27%, reflecting the enhanced robustness and adaptability of the proposed model. Similarly, RoBoTS surpasses Wave-TSVM and TSVM by 2.28% and 2.34%, respectively, showcasing its superior generalization ability across the subclassification tasks of the BreaKHis dataset. In terms of average rank, the proposed RoBoTS achieves the best rank of 1.91, indicating consistent top-tier performance across most subclassification tasks. Among the baselines, TSVM, Wave-TSVM, and RoBoSS-SVM follow with ranks of 4.00, 4.03, and 4.31, respectively, while Pin-GTSVM records a rank of 4.44. SLTSVM (6.03), RVFL (7.84), and the ensemble methods (RF: 8.34, XGB: 9.09) perform less consistently, and SVM ranks the lowest with 9.81. Overall, the results in Table 3.17 demonstrate that the proposed RoBoTS model consistently achieves higher accuracy and superior rank compared to all baseline models. This performance can be attributed to the bounded, smooth, and sparse properties of the RoBoSS loss function, which collectively enhance the model's robustness and stability in histopathological image classification tasks.

- **ADNI dataset:** We now evaluate the performance of the proposed nonlinear RoBoTS model against baseline models on the ADNI dataset, accessible at $adni.loni.usc.edu$. Alzheimer's disease (AD) is a well-known neurodegenerative disorder that gradually leads to memory loss and cognitive decline [93]. The progression of AD is irreversible and is characterized by significant atrophy in the brain's inner regions. Research sug-

gests that early detection and timely intervention can help slow the progression of AD, emphasizing the importance of initiating treatment at the earliest possible stage [94]. The ADNI project was established to investigate various neuroimaging techniques, such as positron emission tomography, magnetic resonance imaging, and other diagnostic methods, particularly focusing on detecting mild cognitive impairment (MCI) as an early stage of AD. The ADNI dataset serves as a valuable resource for developing and evaluating machine learning models for AD diagnosis. In this study, we use the same feature extraction pipeline as described in [95]. The dataset includes three classification cases: control normal (CN) versus AD, CN versus MCI, and MCI versus AD. Table 3.18 presents the performance comparison of the proposed RoBoTS model against various baseline models for these three classification tasks. The proposed RoBoTS model demonstrates superior performance across all cases, achieving the highest average accuracy of 84.68%, outperforming all competing methods. Among the baseline approaches, dRVFL attains the second-highest average accuracy of 82.92%, followed by Wave-TSVM with 82.40% and TSVM with 81.71%. Pin-GTSVM, RoBoSS-SVM, and SLTSVM achieve comparable accuracies of 81.02%, 81.50%, and 80.91%, respectively. Ensemble and traditional models such as RVFL (81.60%), RF (77.57%), XGB (77.42%), and SVM (74.89%) perform relatively lower. For the CN versus AD classification, RoBoTS achieves the highest accuracy of 94.23%, outperforming all baselines, including the next best-performing model, Wave-TSVM, which attains 93.27%. Similarly, for CN versus MCI, RoBoTS records an accuracy of 78.98%, surpassing TSVM (77.07%) and dRVFL (76.43%). In the MCI versus AD case, RoBoTS achieves 80.82%, again outperforming dRVFL (79.19%) and Wave-TSVM (78.77%). The consistent improvements across all three cases highlight the robustness and adaptability of the proposed RoBoTS model in

handling the subtle variations and overlapping patterns inherent in Alzheimer's disease progression. In terms of average rank, RoBoTS secures the top position with a rank of 1.00, reflecting its consistent superiority across all classification tasks. The dRVFL model ranks second with 2.67, followed by Wave-TSVM (4.00) and TSVM (5.17). Pin-GTSVM, RoBoSS-SVM, and SLTSVM follow with ranks of 5.67, 5.67, and 6.67, respectively. Traditional and ensemble methods, including RVFL (6.33), SVM (9.17), RF (9.67), and XGB (10.00), exhibit comparatively weaker consistency and performance. Overall, the results in Table 3.18 clearly demonstrate that the proposed RoBoTS model not only achieves the highest accuracy across all classification scenarios but also consistently ranks first among all compared methods. These findings reaffirm the effectiveness of the RoBoSS loss function and the robustness of the proposed RoBoTS framework in modeling the complex decision boundaries associated with AD classification.

Table 3.19: Classification accuracy comparison of models on noise-injected datasets. The noise level indicates the percentage of label corruption. The proposed RoBoTS model demonstrates superior robustness to increasing noise levels.

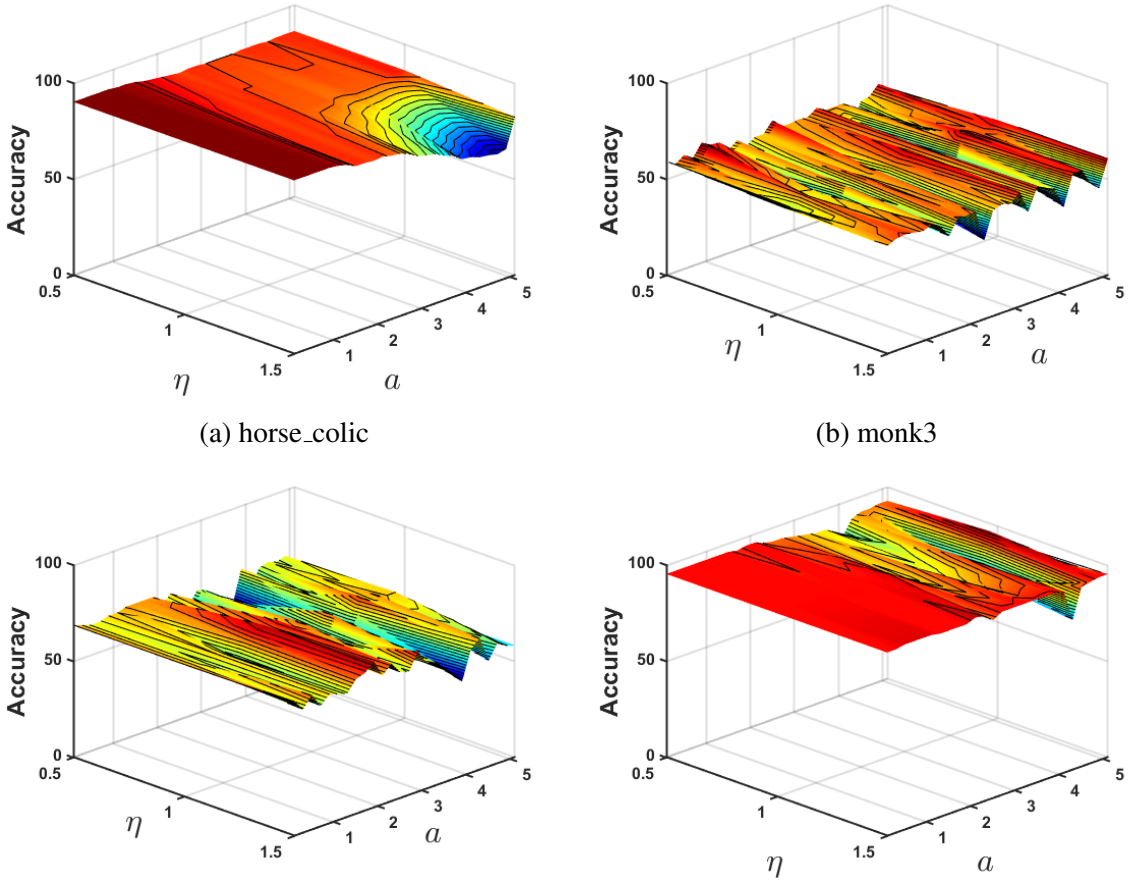
Dataset	Noise (%)	SVM [6]	RF [91]	XGB [92]	RVFL [16]	TSVM [10]	Pin-GTSVM [88]	SLTSVM [89]	Wave-TSVM [67]	RoBoSS-SVM [90]	RoBoTS†
	5	97.2	95.8	94.9	93.7	97.9	97.6	97.3	97.5	96.8	98.6
	10	95.1	93.4	92.8	91.6	96.2	95.8	95.1	95.3	94.7	97.9
breast_cancer_wisc	20	91.3	88.9	88.1	86.4	93.2	92.4	91.7	92.1	91.0	96.5
	30	86.8	83.7	82.4	80.9	89.1	88.3	87.6	88.0	87.1	95.1
	Avg. Acc.	92.6	90.5	89.6	88.1	<u>94.1</u>	93.5	92.9	93.2	92.4	**96.8**
	5	81.9	83.5	82.7	90.6	94.3	87.9	92.5	91.6	94.0	97.8
	10	79.4	80.2	79.5	87.3	92.1	85.2	89.6	88.8	91.2	96.4
credit_approval	20	75.6	76.8	75.9	83.4	88.3	81.5	85.1	84.2	87.3	94.8
	30	71.8	72.6	72.1	79.2	83.7	77.4	81.6	80.8	83.5	92.7
	Avg. Acc.	77.2	78.3	77.6	85.1	<u>89.6</u>	83.0	87.2	86.4	89.0	**95.4**
	5	74.3	86.2	84.3	85.4	86.6	85.7	84.6	82.9	90.5	88.8
	10	71.6	83.4	81.5	82.7	83.9	83.1	81.8	80.3	87.7	87.1
horse_colic	20	67.9	79.1	77.0	78.6	80.2	79.5	78.2	76.8	84.1	85.4
	30	63.8	75.4	72.9	74.2	76.3	75.8	74.5	73.1	80.6	83.2
	Avg. Acc.	69.4	81.0	78.9	80.2	81.8	81.0	79.8	78.3	<u>85.7</u>	**86.1**
	5	94.1	94.5	94.3	94.1	95.8	94.2	95.6	95.4	95.9	97.6
	10	91.6	91.8	91.5	91.2	93.7	92.1	93.3	93.1	93.5	96.8
led7digit-0-2-4-5-6-7-8-9_vs_1	20	88.4	88.7	88.5	88.0	90.6	89.0	90.1	89.8	90.3	95.3
	30	84.2	84.6	84.3	83.8	86.9	85.2	86.0	85.6	86.4	93.8
	Avg. Acc.	89.6	89.9	89.6	89.3	<u>91.8</u>	90.1	91.3	90.9	91.5	**95.9**
	5	51.8	49.5	50.3	54.8	53.9	55.7	53.9	54.0	60.4	55.9
	10	49.3	47.0	47.8	52.3	51.4	53.1	51.3	51.5	58.0	54.6
monk1	20	46.1	44.0	44.7	49.1	48.4	50.0	48.3	48.4	54.9	53.0
	30	43.0	41.2	41.8	46.0	45.3	47.0	45.2	45.3	52.1	51.8
	Avg. Acc.	47.6	45.4	46.2	50.5	49.8	51.5	49.7	49.8	**56.4**	<u>53.8</u>

† denotes the proposed model. Best and second-best results are bold and underlined, respectively.

3.9.4 Experimental Results on Noisy Datasets

To further assess the robustness of the proposed RoBoTS model in noisy environments, we conduct additional experiments on noise-injected datasets. This analysis provides empirical evidence of the model's stability under label perturbations and supports its claimed resilience to noise. In the experiment, label noise is simulated by randomly flipping a specified percentage of class labels in each dataset. Four levels of noise, 5%, 10%, 20%, and 30%, are introduced to evaluate how progressively corrupted labels affect model performance. Five datasets are selected to cover a diverse range of data distributions: breast_cancer_wisc, credit_approval, horse_colic, led7digit-0-2-4-5-6-7-8-9_vs_1, and monk1. These datasets are chosen strategically to analyze different scenarios of model behavior under noise. Specifically, breast_cancer_wisc represents a case where the proposed RoBoTS model performs comparably to the best baseline methods under clean conditions (a tie). The credit_approval and led7digit-0-2-4-5-6-7-8-9_vs_1 datasets are selected as representative cases where RoBoTS achieves the best overall performance, allowing us to examine its robustness when it already dominates under clean data. Conversely, horse_colic and monk1 are included as challenging cases where RoBoTS does not yield the best accuracy on noise-free data, enabling an analysis of how effectively the model sustains performance when its initial advantage is limited. Table 3.19 presents the classification accuracy of all compared models across the four noise levels. The results reveal that the proposed RoBoTS model consistently demonstrates strong resilience to increasing noise, exhibiting a slower rate of accuracy degradation compared to traditional, ensemble, and kernel-based baselines. Across the five evaluated datasets, RoBoTS achieves the best performance on four datasets and the second-best on one, confirming its superior noise tolerance. Specifically, on the breast_cancer_wisc dataset, RoBoTS attains the highest average accuracy of 96.8%, outperforming the next best model, TSVM, which achieves 94.1%. On the credit_approval

dataset, RoBoTS reaches an average accuracy of 95.4%, outperforming all other models by a clear margin. Likewise, on the led7digit-0-2-4-5-6-7-8-9_vs_1 dataset, RoBoTS secures the highest average accuracy of 95.9%, maintaining stable performance even under 30% label corruption. For the horse_colic dataset, RoBoTS again achieves the top average accuracy of 86.1%, marginally surpassing the second-best baseline (85.7%). On the more challenging monk1 dataset, the proposed model achieves the second-best average accuracy of 53.8%, just below RoBoSS-SVM (56.4%). Importantly, in both cases, RoBoTS demonstrates a gentler decline in accuracy as noise levels increase, indicating that its degradation is less severe than that of other baselines. Overall, these results confirm that the proposed RoBoTS model is not only effective on clean data but also exhibits enhanced robustness to noisy and corrupted labels.



(a) horse_colic

(b) monk3

(c) congressional_voting

(d) yeast-0-3-5-9_vs_7-8

Figure 3.5: Sensitivity analysis of the RoBoTS model with respect to the loss hyperparameters a and η, presented through 3D surface plots for four datasets: (a) horse_colic, (b) monk3, (c) congressional_voting, and (d) yeast-0-3-5-9_vs_7-8. The color scale represents accuracy values, with warmer colors (yellow–red) indicating higher accuracy and cooler colors (blue–green) indicating lower accuracy.

3.9.5 Sensitivity Analysis of the RoBoSS Loss Hyperparameters

In this subsection, we evaluate the sensitivity of the proposed RoBoTS model to its loss function hyperparameters: the bounding parameter (η) and the shape parameter (a). These hyperparameters play a pivotal role in shaping the behavior of the RoBoSS loss function by regulating the maximum penalty imposed and the severity of penalization for misclassified samples. Understanding their impact provides critical insights for optimizing model performance across diverse datasets. Figure 3.5 presents the sensitivity analysis results for four

benchmark datasets: (a) horse_colic, (b) monk3, (c) congressional_voting, and (d) yeast-0-3-5-9_vs_7-8. The 3D surface plots illustrate the variations in model accuracy as a and η are adjusted, while all other hyperparameters are fixed at their optimal values. The color scale represents the corresponding accuracy values, where warmer colors (yellow–red) indicate higher accuracy and cooler colors (blue–green) indicate lower accuracy. The subfigures reveal dataset-specific patterns in how a and η affect accuracy. In Figure 3.5a (horse_colic), moderate values of a and smaller values of η deliver optimal performance, striking a balance between penalizing misclassifications and avoiding overfitting. As a or η increases, accuracy drops, highlighting the dataset's sensitivity to hyperparameter tuning. In Figure 3.5b (monk3), the model shows robustness across a wider range of a and η, with peak accuracy achieved at higher values of both parameters. For Fig. 3.5c (congressional_voting), the surface plot underscores the importance of smaller η values paired with moderate a values. Larger η values lead to a rapid decline in accuracy, likely due to the presence of noise or outliers in the dataset, making robustness a critical factor for this classification task. Finally, in Figure 3.5d (yeast-0-3-5-9_vs_7-8), accuracy is highly sensitive to changes in a and η. Moderate values of both parameters (a and η) yield optimal results. The sharp decline in accuracy beyond these values indicates the dataset's vulnerability to overfitting when the loss function penalizes misclassifications too aggressively. These observations emphasize the dataset-dependent behavior of the RoBoSS loss hyperparameters. Smaller values of a and η generally enhance robustness to noise and outliers, as seen in congressional_voting and horse_colic. Conversely, datasets like monk3 and yeast-0-3-5-9_vs_7-8 benefit from larger values, leveraging the model's ability to delineate complex decision boundaries. This analysis highlights the importance of carefully tuning a and η to balance robustness and sensitivity. The findings validate the flexibility of the RoBoSS loss function, demonstrating its adaptability across diverse datasets.

3.9.6 Effectiveness of the Proposed RoBoTS Model

The outstanding performance of the proposed RoBoTS model on the UCI, KEEL, BreaKHis, and ADNI datasets can be attributed to the unique properties of the RoBoSS loss function. Its bounded nature provides resilience to outliers, ensuring that extreme values do not disproportionately influence the model's learning process. Additionally, the sparsity of the loss function encourages the use of only the most relevant samples during training, resulting in simpler models. The smoothness of the RoBoSS loss further facilitates efficient optimization, enabling stable and rapid convergence. Additionally, the integration of a regularization term aligned with the principle of structural risk minimization (SRM) enhances the model's capacity to generalize by controlling complexity and mitigating overfitting. Beyond empirical performance, the theoretical soundness of the proposed model is reinforced through a generalization error bound derived using the framework of Rademacher complexity. This bound provides a formal guarantee on the expected risk of the classifier, substantiating the model's generalization capability under the RoBoSS loss. The empirical findings from Tables 3.12 to 3.19 clearly demonstrate that the proposed RoBoTS model significantly outperforms baseline methods across a wide range of datasets, including benchmark tabular data (KEEL and UCI) and complex biomedical datasets (BreaKHis and ADNI). This establishes RoBoTS as a robust and effective tool for real-world classification tasks, particularly in the biomedical domain, where accurate and reliable classification is paramount.

3.10 Chapter Summary

This chapter introduced the RoBoSS loss function as a bounded, sparse, smooth, and robust loss for supervised learning. The RoBoSS-SVM formulation showed how the loss can be integrated into a standard SVM framework and optimized efficiently with a Nesterov

accelerated gradient strategy. The chapter then extended the same loss principle to twin support vector machines through RoBoTS and inverse-free RoBoTS, demonstrating how bounded loss design can improve robustness while preserving computational efficiency. The experimental evidence across benchmark, noisy, outlier-contaminated, and biomedical datasets supports the effectiveness of RoBoSS-based learning models under challenging data conditions.

Chapter 4

Advancing Supervised Learning with the Wave Loss Function: A Robust and Smooth Approach

This chapter develops the Wave loss function, an asymmetric loss for robust and smooth supervised classification. The choice of loss function strongly influences the learned model because it determines how margin violations, correctly classified samples, noise, and outliers are penalized during optimization. Existing losses often provide only part of the desired behavior: some are convex but unbounded and noise-sensitive, while others are bounded but non-smooth or difficult to optimize. The Wave loss is designed to combine boundedness, smoothness, outlier robustness, and noise insensitivity in a single formulation, while retaining a classification-calibrated behavior. Building on this loss, the chapter develops Wave-SVM and Wave-TSVM by integrating the Wave loss into the least-squares settings of support vector machine (SVM) and twin support vector machine (TSVM), respectively. Wave-SVM is optimized using adaptive moment estimation (Adam), whereas Wave-TSVM is solved through an efficient iterative scheme. The chapter validates these models on UCI, KEEL, noisy benchmark datasets, and the ADNI biomedical dataset, showing that Wave-

loss-based learning improves predictive performance under data uncertainty.

4.1 Introduction and Motivation

In the realm of machine learning, supervised learning stands as a cornerstone, which empowers to build models that can make accurate predictions and classifications. A critical component within this paradigm is the choice of an appropriate loss function, which plays a crucial role in guiding the training process. The loss function quantifies the disparity between predicted outcomes and true labels, providing an evident indication of a model's performance [96]. Its role extends beyond mere evaluation; it fundamentally shapes the model's learning trajectory, influencing its generalization ability, and aligning its predictions with the intrinsic goals of the task.

Hyperplane-based classifiers such as SVM and TSVM provide the modeling setting for this chapter. For $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{l}$, with $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$, a regularized soft-margin hyperplane learner can be written compactly as

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \mathfrak{L}(1 - y_i(w^\intercal x_i + b))\,, \tag{4.1}$$

where $\mathfrak{L}(\cdot)$ is the classification loss and $C > 0$ balances regularization with empirical loss. TSVM follows the same margin-based philosophy but learns two non-parallel hyperplanes through smaller subproblems, making it attractive for efficient classification [10, 97]. This chapter uses SVM and TSVM only as host models for studying the effect of a new robust loss.

Existing classification losses used as reference points in this thesis include hinge, pinball, ramp, squared hinge, smooth pinball, LINEX, and related bounded variants; their formulations and properties are summarized in Table 2.2. The main observation needed here

is that unbounded losses can remain sensitive to severe outliers, hard-truncated bounded losses are typically non-smooth, and smooth bounded losses do not always provide the desired noise-insensitive behavior around the decision boundary. This motivates a loss that is bounded and smooth while retaining enough asymmetry to adapt the penalty assigned to different margin deviations.

Taking inspiration from prior research endeavors, this chapter develops a novel asymmetric loss function, referred to as the wave loss function (see Figure 4.1). It is precisely engineered to exhibit robustness against outliers, insensitivity to noise, and a propensity for smoothness. Subsequently, by integrating the proposed wave loss into the least squares framework of SVM and TSVM, we present two novel models, namely Wave-SVM and Wave-TSVM. The optimization problem associated with the Wave-SVM is effectively addressed through the utilization of the adaptive moment estimation (Adam) algorithm. Further, to solve the optimization problems of the Wave-TSVM an efficient iterative algorithm is utilized. The major contributions of this chapter can be outlined as follows:

- We propose a novel asymmetric loss function named wave loss, designed to exhibit robustness against outliers, insensitivity to noise, and smooth characteristics. Further, we delve into the theoretical aspect of the wave loss function and validate its capability to maintain a vital classification-calibrated property.

- We amalgamate the proposed wave loss function into the least squares setting of SVM and TSVM and introduce two novel robust and smooth models, termed Wave-SVM and Wave-TSVM, respectively.

- We address the optimization problem inherent to Wave-SVM by employing the Adam algorithm, known for its lower memory requirements and efficacy in handling large-scale problems. To our knowledge, this is the first time Adam has been used to solve

an SVM problem. Further, we utilized an efficient iterative algorithm to solve the optimization problems of Wave-TSVM.

- We perform comprehensive numerical experiments using benchmark UCI and KEEL datasets (with and without feature noise) from various domains. The outcomes vividly showcase the outstanding performance of the proposed Wave-SVM and Wave-TSVM when contrasted with baseline models.

- To demonstrate the superiority of the proposed Wave-SVM and Wave-TSVM models in the biomedical realm, we conducted experiments using the ADNI dataset. These empirical investigations furnish substantial evidence of the proposed models applicability in real-world medical scenarios.

The remainder of this chapter presents the Wave loss function, develops the Wave-SVM and Wave-TSVM formulations, describes the optimization procedures and computational complexity, and then reports the numerical and statistical analyses.

4.2 Wave Loss and Proposed Models

In this section, we introduce a groundbreaking asymmetric loss function named wave loss and elucidate its distinctive characteristics. Subsequently, we amalgamate the proposed wave loss into the frameworks of SVM and TSVM within the least squares setting and propose two novel models, namely Wave-SVM and Wave-TSVM. We also analyze the computational complexity associated with the proposed algorithms.

4.2.1 Wave Loss Function

We present a substantial advancement within the domain of supervised learning: a novel asymmetric loss function that embodies robustness against outliers, insensitivity to noise,

and smoothness characteristics. This novel loss function, referred to as the wave loss function, is visually depicted in Figure 4.1. The mathematical expression of the proposed wave loss function is articulated as follows:

$$\mathfrak{L}_{wave}(u) = \frac{1}{\lambda}\left(1 - \frac{1}{1 + \lambda u^2 e^{au}}\right), \ \forall \ u \in \mathbb{R}, \tag{4.2}$$

where $a \in \mathbb{R}$ is shape parameter and $\lambda \in \mathbb{R}^+$ is bounding parameter. The wave loss function (4.2), as delineated in this work, showcases the following inherent properties:

- It is robust to outliers and insensitive to noise. As it bound the loss to $1/\lambda$, which gives robustness to outliers, and it also gives loss to samples with $u \leq 0$, which produces insensitivity towards noise.

- It is non-convex, smooth, and bounded.

- It is infinitely differentiable and hence continuous for all $u \in \mathbb{R}$.

- For given value of λ, when $a \to +\infty$, the wave loss (4.2) converges point-wise to the "$0 - 1/\lambda$" loss which is defined as

$$\mathfrak{L}_{0-1}(u) = \begin{cases} \frac{1}{\lambda}, & u > 0, \\ 0, & u \leq 0. \end{cases}$$

  Furthermore, when $\lambda = 1$ it converges to "$0 - 1$" loss.

- It has two advantageous parameters: shape parameter (a) determines the loss function's shape, and bounding parameter (λ) sets thresholds for loss values.

4.2.2 Smoothness of the Wave Loss Function

Mathematical justification for the smoothness of wave loss function:

The proposed wave loss function is formulated with continuous and exponential components, ensuring its overall differentiability. However, to alleviate concerns regarding its differentiability, we provide a rigorous mathematical justification. Specifically, we demonstrate that for any arbitrary $u \in \mathbb{R}$, the left-hand derivative ($L.H.D.$) and right-hand derivative ($R.H.D.$) are equivalent, affirming the function's smoothness and analytical tractability.

Proof. The left-hand and right-hand derivatives of a function $\mathfrak{L}(u)$ at an arbitrary point $u \in \mathbb{R}$ are defined as: $\lim_{h\to 0^-} \frac{\mathfrak{L}(u+h)-\mathfrak{L}(u)}{h}$ and $\lim_{h\to 0^+} \frac{\mathfrak{L}(u+h)-\mathfrak{L}(u)}{h}$, respectively. First, we compute the left-hand derivative of the wave loss function $\mathfrak{L}_{wave}(u)$:

$$
\begin{aligned}
L.H.D. &= \lim_{h\to 0^-} \frac{\mathfrak{L}_{wave}(u+h) - \mathfrak{L}_{wave}(u)}{h} \\
&= \lim_{h\to 0^-} \frac{\frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda(u+h)^2 e^{a(u+h)}}\right) - \frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda u^2 e^{au}}\right)}{h} \\
&= \lim_{h\to 0^-} \frac{1}{\lambda h}\left(\frac{1}{1+\lambda u^2 e^{au}} - \frac{1}{1+\lambda(u+h)^2 e^{a(u+h)}}\right) \\
&= \lim_{h\to 0^-} \frac{(u+h)^2 e^{a(u+h)} - u^2 e^{au}}{h\left(1+\lambda u^2 e^{au}\right)\left(1+\lambda(u+h)^2 e^{a(u+h)}\right)} \qquad (\frac{0}{0} \text{ form}) \\
&= \lim_{h\to 0^-} \frac{a(u+h)^2 e^{a(u+h)} + 2(u+h)e^{a(u+h)}}{(1+\lambda u^2 e^{au})[1+\lambda(u+h)^2 e^{a(u+h)} + \lambda h(u+h)e^{a(u+h)}(a(u+h)+2)]} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Using L'hospital rule)} \\
&= \frac{ue^{au}(au+2)}{(1+\lambda u^2 e^{au})^2}.
\end{aligned}
$$

Now, we compute the right-hand derivative of the wave loss function $\mathfrak{L}_{wave}(u)$:

$$
R.H.D. = \lim_{h\to 0^+} \frac{\mathfrak{L}_{wave}(u+h) - \mathfrak{L}_{wave}(u)}{h}.
$$

In a similar manner, we can find that the right-hand derivative is also $\frac{ue^{au}(au+2)}{(1+\lambda u^2 e^{au})^2}$. Since, the

left-hand derivative and right-hand derivative of the wave loss function are equal for any arbitrary $u \in \mathbb{R}$. Therefore, we have shown that the wave loss function is smooth in the whole domain. □

It is crucial to note that the existing bounded loss functions set a strict limit on the maximum allowable loss for data points with substantial deviations. This effectively prevents noise and outliers from exerting excessive influence, thus enhancing the model's robustness. However, the majority of the existing loss functions use a hard truncation strategy to bound the loss, which makes them non-smooth. In contrast, the proposed wave loss function is smooth and bounded simultaneously (refer to Figure 4.1). In particular, Figures 4.1a, 4.1b, 4.1c, and 4.1d illustrate how the loss profile changes for representative values of the shape parameter. By adjusting the bounding parameter λ, the wave loss function smoothly bounds the loss to a predefined value. The primary benefit of the smoothness characteristic lies in its facilitation of the application of gradient-based optimization algorithms. This property ensures the existence of well-defined gradients, enabling the utilization of rapid and reliable optimization techniques.

4.2.3 Theoretical Analysis of the Wave Loss Function

In this subsection, we analyze the theoretical characteristics inherent to the proposed wave loss function and provide evidence to demonstrate that the wave loss function exhibits a crucial classification-calibrated property. Bartlett et al. [26] introduced the concept of classification-calibration to assess the statistical effectiveness of loss functions in the context of classification learning. It ensures that the model's predicted probabilities reflect the true likelihood of an event. For more details, readers can refer to [26, 98]. This attribute holds crucial importance in enhancing our understanding of the performance dynamics of the wave loss function within classification tasks.

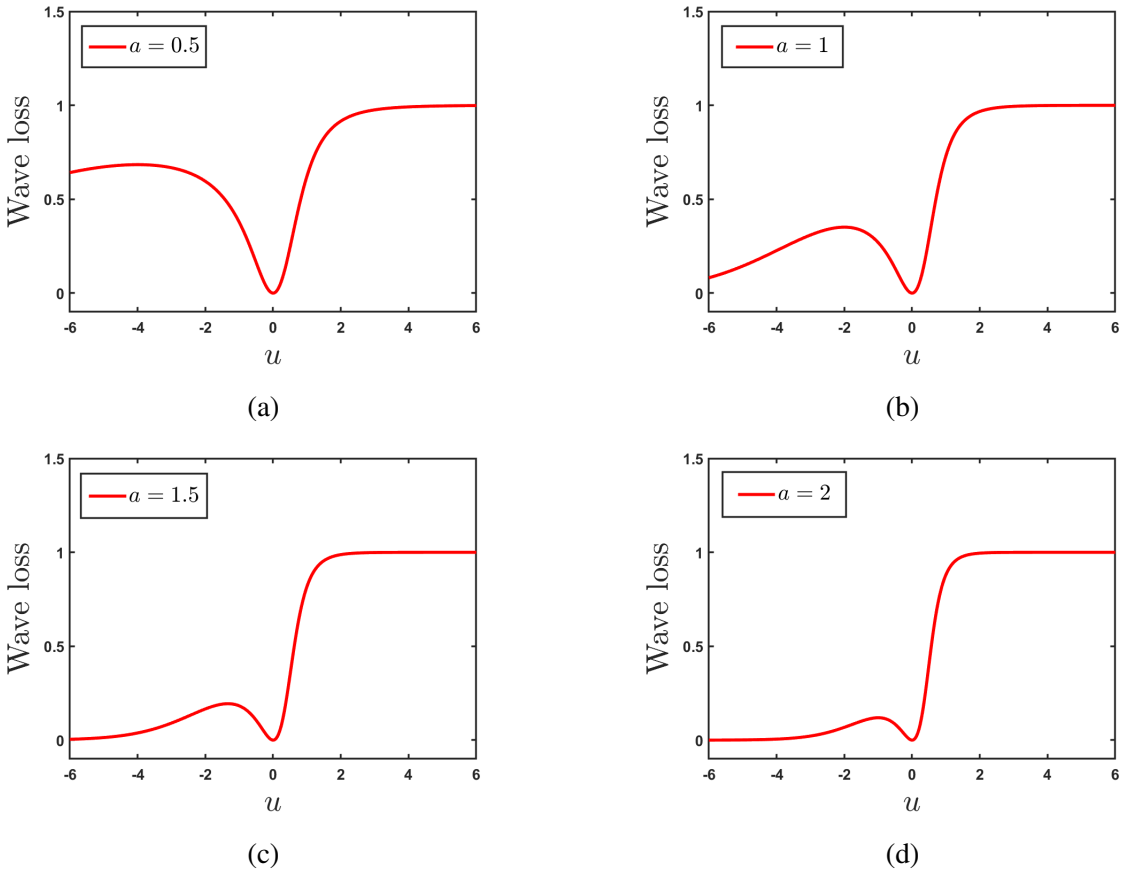


Figure 4.1: Illustration of wave loss function for fixed $\lambda = 1$ and different values of a. Subfigures (a), (b), (c), and (d) demonstrate that the value of a controls the strength of the penalty for correctly classified and misclassified samples.

Consider that the training data $\mathcal{D} = \{x_i, y_i\}_{i=1}^{l}$ is independently sampled from a probability distribution P. The distribution P is defined over a combined space of inputs and corresponding labels, where the input space X is a subset of $\mathbb{R}^n$ and the label space Y consists of two possible labels: -1 and 1. In this context, the main objective is to develop a binary classifier C, which takes inputs from the space X and assigns them to one of the two labels in Y. The central goal of this classification problem is to design a classifier in a way that minimizes the associated error. The risk associated with a specific classifier C is quantified by the following mathematical expression:

$$\mathsf{R}(\mathsf{C}) = \int_X \mathsf{P}(y \neq \mathsf{C}(x)|x) d\mathsf{P}_X.$$

Here $\mathsf{P}(y|x)$ signifies the probability distribution of the label y given an input x, and $d\mathsf{P}_X$ represents the marginal distribution of the input x according to the distribution P. Moreover, the conditional distribution $\mathsf{P}(y|x)$ is a binary distribution, implying it is determined by the probabilities $\texttt{Prob}(y = 1|x)$ and $\texttt{Prob}(y = -1|x)$. To ease in writing, we further utilize $\mathsf{P}(x)$ and $1-\mathsf{P}(x)$ to represent $\texttt{Prob}(y = 1|x)$ and $\texttt{Prob}(y = -1|x)$, respectively. In simpler terms, the objective is to construct a classifier that accurately assigns labels to inputs while minimizing the overall error.

Now, the Bayes classifier, for $\mathsf{P}(x) \neq 1/2$, is defined as follows:

$$f_{\mathsf{C}}(x) = \begin{cases} -1, & \mathsf{P}(x) < 1/2, \\ 1, & \mathsf{P}(x) > 1/2. \end{cases} \tag{4.3}$$

It can be verified that the Bayes classifier effectively minimizes the classification error. The following expression mathematically expresses the previous statement:

$$f_{\mathsf{C}} = \arg \min_{\mathsf{C}:X \to Y} \mathsf{R}(\mathsf{C}).$$

Now, for any given loss function $\mathfrak{L}$, the expected error associated with a classifier $f{:}X \to \mathbb{R}$ is formulated as follows:

$$\mathsf{R}_{\mathfrak{L},\mathsf{P}}(f) = \int_{X\times Y} \mathfrak{L}(1 - yf(x))d\mathsf{P}. \tag{4.4}$$

The function $f_{\mathfrak{L},\mathsf{P}}$, which minimizes the expected error over all measurable functions, is formulated as:

$$f_{\mathfrak{L},\mathsf{P}}(x) = \arg\min_{f(x)\in\mathbb{R}} \int_Y \mathfrak{L}\left(1 - yf(x)\right) d\mathsf{P}(y|x),\ \forall x \in X. \tag{4.5}$$

Subsequently, for the proposed wave loss $\mathfrak{L}_{wave}(\cdot)$, we can derive Theorem 4.2.1, providing evidence of the classification-calibrated nature [26] of the wave loss function. This characteristic, a valuable aspect of a loss function, ensures that the minimizer of expected error aligns with the sign of the Bayes classifier.

Theorem 4.2.1. *The proposed loss function $\mathfrak{L}_{wave}(u)$ is classification-calibrated, i.e., $f_{\mathfrak{L}_{wave},\mathsf{P}}$ has the same sign as the Bayes classifier.*

Proof. After simple computation, we arrive at the following outcome:

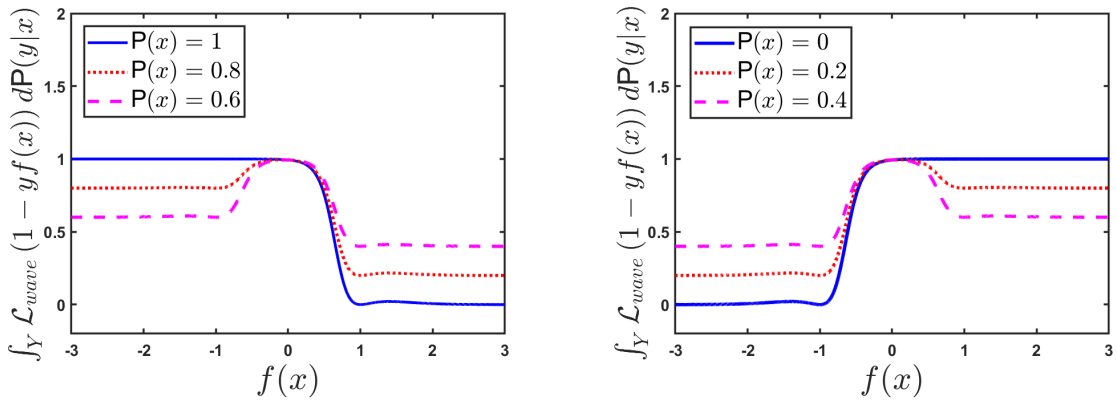
$$\begin{aligned}
&\int_Y \mathfrak{L}_{wave}\left(1 - yf(x)\right) d\mathsf{P}(y|x) \\
&= \mathfrak{L}_{wave}(1 - f(x))\mathsf{P}(x) + \mathfrak{L}_{wave}(1 + f(x))(1 - \mathsf{P}(x)) \\
&= \frac{1}{\lambda}\left(1 - \frac{1}{1 + \lambda(1 - f(x))^2 e^{a(1-f(x))}}\right)\mathsf{P}(x) \\
&\quad + \frac{1}{\lambda}\left(1 - \frac{1}{1 + \lambda(1 + f(x))^2 e^{a(1+f(x))}}\right)(1 - \mathsf{P}(x)).
\end{aligned}$$

In Figures 4.2a and 4.2b, the graphical representations of $\int_Y \mathfrak{L}_{wave}\left(1 - yf(x)\right) d\mathsf{P}(y|x)$ are depicted as functions of $f(x)$, considering the cases when $\mathsf{P}(x)$ is greater than $1/2$ and when it is less than $1/2$, respectively. As discernible from Figure 4.2, when $\mathsf{P}(x)$ exceeds $1/2$, the

minimum value of $\int_Y \mathfrak{L}_{wave}\left(1 - yf(x)\right) d\mathsf{P}(y|x)$ is achieved for a positive value of $f(x)$. Conversely, when $\mathsf{P}(x)$ is less than $1/2$, the minimum value corresponds to a negative value of $f(x)$.

Hence, it becomes evident from the patterns observed in Figures 4.2a and 4.2b that the proposed loss function $\mathfrak{L}_{wave}(u)$ holds the characteristic of being classification-calibrated.

□



(a) (b)

Figure 4.2: Illustrate the plot of $\int_Y \mathfrak{L}_{wave}\left(1 - yf(x)\right) d\mathsf{P}(y|x)$ over $f(x)$ with varying values of $\mathsf{P}(x)$. (a) Depict the case where $\mathsf{P}(x)$ is greater than $1/2$, and (b) illustrate the scenario where $\mathsf{P}(x)$ less than $1/2$.

4.2.4 Formulation of Wave-SVM

Through the amalgamation of the proposed wave loss function into the least squares setting of SVM, we construct a novel support vector classifier for large-scale problems that manifest robustness against outliers, insensitivity to noise, and smoothness characteristics. This advanced classifier is denoted as the Wave-SVM. For simplification, throughout the Wave-SVM formulation, we use the terminology w for $[w^\intercal, b]$ and x_i for $[x_i, 1]^\intercal$. The for-

mulation of non-linear Wave-SVM is given as:

$$\begin{aligned} \underset{w,\xi}{min} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\xi_i^2 e^{a\xi_i}}\right), \\ \text{subject to} \quad & y_i\left(w^\intercal\phi(x_i)\right) = 1 - \xi_i,\ \forall\, i = 1, 2, \ldots, l, \end{aligned} \tag{4.6}$$

where $C > 0$ is the regularization parameter, which determines the trade-off between margin and the penalty, λ and a are the parameters of the wave loss function, and $\phi(\cdot)$ is feature mapping associated with the kernel function. The characteristics of various SVM models, along with the Wave-SVM, are presented in Table 4.1, which showcases that the proposed Wave-SVM possesses most of the desirable attributes required for a better classifier.

Table 4.1: Comparative Analysis of SVM Models Utilizing Different Loss Functions.

Models ↓ \Characteristics →	Robust to outliers	Insensitive to noise	Sparse	Convex	Bounded	Smooth	Fast
C-SVM [6]	✗	✗	✓	✓	✗	✗	✗
Pin-SVM [27]	✗	✓	✗	✓	✗	✗	✗
LINEX-SVM [31]	✗	✓	✗	✓	✗	✓	✓
QTLS [69]	✗	✓	✗	✗	✗	✓	✓
FP-SVM [99]	✗	✓	✗	✓	✗	✗	✗
Wave-SVM (Proposed)	✓	✓	✗	✗	✓	✓	✓

The dual problem of the Wave-SVM is challenging to optimize due to the non-convexity of the wave loss function. However, the inherent smoothness of the Wave-SVM permits us to utilize a gradient-based algorithm to optimize the model. Gradient-based optimization with Wave-SVM offers several advantages. It facilitates faster convergence during training, as gradient-based methods generally converge at a faster rate than quadratic programming solvers [59]. In this chapter, we adopt the Adam [23] optimization technique for solving the Wave-SVM. It is an improved version of stochastic gradient descent with adaptive learning rates. This is the first time the Adam algorithm has been used to solve an SVM model. The Adam algorithm converges more quickly and stays stable during learning due to the adjustable learning rates. It combines the benefits of two other popular optimization algorithms: AdaGrad [60] and RMSProp [100]. The core idea behind Adam involves the

computation of a progressively diminishing average for previous gradients and the squared gradients of the weights. Subsequently, based on these estimations, the algorithm establishes an adaptive learning rate for each weight parameter. It is important to note that the Adam algorithm only requires the gradient of the unconstrained optimization problem, and thus no modification was made to the Adam algorithm for its application to Wave-SVM.

4.2.4.1 Adam for Linear Wave-SVM

For linear Wave-SVM, put $\phi(x) = x$ in equation (4.6), the linear Wave-SVM is defined as:

$$\begin{aligned} \min_{w} f(w) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\xi_i^2 e^{a\xi_i}}\right), \\ \text{subject to} \quad & y_i\left(w^{\intercal}x_i\right) = 1 - \xi_i,\ \forall\, i = 1, 2, \ldots, l. \end{aligned} \tag{4.7}$$

At each iteration t, k samples are chosen, and at that time the value of gradient, exponentially decaying averages of past gradients (first moment), and past squared gradients (second moment) are calculated using (4.8), (4.9), and (4.10), respectively as follows:

$$g_t = \nabla f(w) = w - \sum_{i=1}^{k}\frac{C y_i x_i \xi_i\left(2+\xi_i\right)\exp\left(a\xi_i\right)}{\left[1+\lambda\xi_i^2\exp\left(a\xi_i\right)\right]^2}, \tag{4.8}$$

$$m_t = \beta_1 m_{t-1} + \left(1-\beta_1\right)\nabla f(w_t), \tag{4.9}$$

$$v_t = \beta_2 v_{t-1} + \left(1-\beta_2\right)\nabla f(w_t)^2, \tag{4.10}$$

where β_1 and β_2 are exponential decay rates for the first and second moment estimates, usually set to 0.9 and 0.999, respectively. Then, we compute the bias-corrected first and

second moment estimates using (4.11) and (4.12), respectively.

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)}, \tag{4.11}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)}. \tag{4.12}$$

In the end, the parameter w is updated as:

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{4.13}$$

where α is the learning rate, and ϵ is a small constant used to avoid division by zero. Once the optimal w is achieved, the prediction regarding the label of the unknown sample x can be made using the decision function as:

$$\hat{y} = \text{sign}(f(x)) = \text{sign}\left(w^{\intercal} x\right). \tag{4.14}$$

4.2.4.2 Adam for Nonlinear Wave-SVM

The determination of the dual problem in the proposed Wave-SVM poses challenges that limit the applicability of kernel methods. In this case, to enhance the capacity of Wave-SVM for non-linear adaptation, we use the representer theorem [70], which allows us to express w in equation (4.6) as follows:

$$w = \sum_{j=1}^{l} \gamma_j \phi(x_j), \tag{4.15}$$

where $\gamma = (\gamma_1, \ldots, \gamma_l)^{\intercal}$ is the local kernel coefficient vector.

Substituting (4.15) into (4.6), we obtain:

$$\min_{\gamma} f(\gamma) = \sum_{i=1}^{l} \sum_{j=1}^{l} \frac{1}{2} \gamma_i \gamma_j \mathcal{K}\left(x_i, x_j\right) + C \sum_{i=1}^{l} \frac{1}{\lambda} \left(1 - \frac{1}{1 + \lambda \xi_i^2 e^{a\xi_i}}\right), \tag{4.16}$$

where $\xi_i = 1 - y_i \left(\sum_{j=1}^{l} \gamma_j \mathcal{K}(x_j, x_i) \right)$, and $\mathcal{K}(x_j, x_i) = (\phi(x_j) \cdot \phi(x_i))$ is the Kernel function.

Algorithm 4.1 Non-linear Wave-SVM

Input:
The dataset: $\{x_i, y_i\}_{i=1}^{l}$, $y_i \in \{-1, 1\}$;
The parameters: Regularization parameter C, wave loss parameters λ and a, mini-batch size k, decay rates β_1 and β_2, learning rate α, constant ϵ, error tolerance η, maximum iteration number T;
Initialize: γ_0 and t;
Output:
The classifiers parameters: γ;
1 : Select k samples $\{x_i, y_i\}_{i=1}^{k}$ uniformly at random.
2 : Computing ξ_i :

$$\xi_i = 1 - y_i \left(\sum_{j=1}^{k} \gamma_j \mathcal{K}(x_j, x_i) \right), \quad i = 1, \ldots, k; \tag{4.17}$$

3 : Compute $\nabla f(\gamma_t)$: (4.18);
4 : Compute m_t': (4.19);
5 : Compute v_t': (4.20);
6 : Compute $\hat{m}_t{}'$: (4.21);
7 : Compute $\hat{v}_t{}'$: (4.22);
8 : Update SVM solution γ_t: (4.23);
9 : Update iteration number: $t = t + 1$.
Until:
$|\gamma_t - \gamma_{t-1}| < \eta$ or $t = T$
Return: γ_t.

In the same way, as in linear Wave-SVM, we use the following formulas to obtain the gradient, exponentially decaying averages of past gradients and past squared gradients, and then update the parameter γ. The Adam algorithm structure for non-linear Wave-SVM is clearly described in Algorithm 4.1.

$$\nabla f(\gamma_t) = \mathcal{K}\gamma - \sum_{i=1}^{k} \frac{C y_i \mathcal{K}_i \xi_i (2 + \xi_i) \exp(a\xi_i)}{\left[1 + \lambda \xi_i^2 \exp(a\xi_i)\right]^2}, \tag{4.18}$$

where $\mathcal{K}$ is the Gaussian Kernel matrix of the randomly selected training dataset and $\mathcal{K}_i$ is

the ith row of the matrix $\mathcal{K}$.

$$m_t' = \beta_1 m_{t-1}' + (1-\beta_1)\nabla f(\gamma_t), \tag{4.19}$$

$$v_t' = \beta_2 v_{t-1}' + (1-\beta_2)\nabla f(\gamma_t)^2, \tag{4.20}$$

$$\hat{m_t}' = \frac{m_t'}{(1-\beta_1^t)}, \tag{4.21}$$

$$\hat{v_t}' = \frac{v_t'}{(1-\beta_2^t)}. \tag{4.22}$$

Eventually, the parameter γ is updated as follows:

$$\gamma_t = \gamma_{t-1} - \alpha \frac{\hat{m_t}'}{\sqrt{\hat{v_t}'} + \epsilon}. \tag{4.23}$$

When the optimal γ is obtained, the following decision function can be utilized to predict the label of a new sample x.

$$\hat{y} = \text{sign}(f(x)) = \text{sign}\left(\sum_{j=1}^{k} \gamma_j \mathcal{K}(x_j, x)\right). \tag{4.24}$$

4.2.5 Formulation of Wave-TSVM

In this subsection, we amalgamate the proposed asymmetric wave loss function into the least squares setting of TSVM and propose a robust and smooth classifier referred to as Wave-TSVM. Let $X_+ = \left(x_1, \ldots, x_{l_+}\right)^\top \in \mathbb{R}^{l_+ \times n}$ and $X_- = \left(x_1, \ldots, x_{l_-}\right)^\top \in \mathbb{R}^{l_- \times n}$ represent matrices containing positive and negative instances, where l_+ and l_- denote the count of positive and negative instances, respectively, and $l = l_+ + l_-$. Further, e_1 and e_2 are vectors of ones of appropriate size, and I is the identity matrix of appropriate size.

4.2.5.1 Linear Wave-TSVM

Given a training dataset $\mathcal{D}$, the goal of Wave-TSVM is to obtain the positive and negative hyperplanes as follows:

$$w_+^\top x + b_+ = 0 \quad \text{and} \quad w_-^\top x + b_- = 0, \tag{4.25}$$

where w_+, $w_- \in \mathbb{R}^n$ and b_+, $b_- \in \mathbb{R}$ are the weight vectors and the bias terms, respectively. To obtain the hyperplanes (4.25), the primal of linear Wave-TSVM are formed as follows:

(Linear Wave-TSVM-1)

$$\begin{aligned}
&\min_{w_+,b_+,\xi^-} \frac{1}{2}\sum_{i=1}^{l_+}\left(w_+^\top x_i + b_+\right)^2 + \frac{1}{2}C_1\left(\|w_+\|_2^2 + b_+^2\right) + C_2\sum_{j=1}^{l_-}\xi_j^- \\
&\text{s.t. } \xi_j^- = \frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1+w_+^\top x_j + b_+\right)^2\exp\{a\left(1+w_+^\top x_j + b_+\right)\}}\right), \quad j = 1,\ldots,l_-,
\end{aligned} \tag{4.26}$$

(Linear Wave-TSVM-2)

$$\begin{aligned}
&\min_{w_-,b_-,\xi^+} \frac{1}{2}\sum_{j=1}^{l_-}\left(w_-^\top x_j + b_-\right)^2 + \frac{1}{2}C_3\left(\|w_-\|_2^2 + b_-^2\right) + C_4\sum_{i=1}^{l_+}\xi_i^+, \\
&\text{s.t. } \xi_i^+ = \frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1-w_-^\top x_i - b_-\right)^2\exp\{a\left(1-w_-^\top x_i - b_-\right)\}}\right), \quad i = 1,\ldots,l_+
\end{aligned} \tag{4.27}$$

where $\xi_+ = \left(\xi_1^+,\ldots,\xi_{l_+}^+\right)^\top \in \mathbb{R}^{l_+}, \xi_- = \left(\xi_1^-,\ldots,\xi_{l_-}^-\right)^\top \in \mathbb{R}^{l_-}$. For the sake of brevity, we solely discuss optimization problem (4.26), noting that optimization problem (4.27) follows a similar structure. The objective function provided in equation (4.26) comprises three distinct terms. More precisely, the initial term aims to minimize the distance between the positive hyperplane and the positive instances. The second term, a regularization term, is incorporated to enhance the generalization performance of the model. The third term

represents the cumulative penalty of all negative samples, leveraging the proposed wave loss function.

Given the non-convex nature of optimization problems (4.26) and (4.27), and utilizing their inherent smoothness, we devise an iterative algorithm to solve them. Initially, we convert (4.26) and (4.27) into vector-matrix form in the following manner:

$$\min_{w_1} P_1(w_1) = \frac{1}{2}\left\|G^\top w_1\right\|_2^2 + \frac{1}{2}C_1\left\|w_1\right\|_2^2 + C_2\mathfrak{L}_1(w_1), \tag{4.28}$$

and

$$\min_{w_2} P_2(w_2) = \frac{1}{2}\left\|H^\top w_2\right\|_2^2 + \frac{1}{2}C_3\left\|w_2\right\|_2^2 + C_4\mathfrak{L}_2(w_2), \tag{4.29}$$

where $\mathfrak{L}_1(w_1) = \sum_{j=1}^{l_-}\frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1+H_j^\top w_1\right)^2\exp\{a\left(1+H_j^\top w_1\right)\}}\right),\ j = 1,\ldots,l_-;\ \mathfrak{L}_2(w_2) = \sum_{i=1}^{l_+}\frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1-G_i^\top w_2\right)^2\exp\{a\left(1-G_i^\top w_2\right)\}}\right),\ i = 1,\ldots,l_+$. G_i is the ith column of matrix G and H_j is the jth column of matrix H. $G = [X_+, e_1]^\top \in \mathbb{R}^{(n+1)\times l_+}$, $H = [X_-, e_2]^\top \in \mathbb{R}^{(n+1)\times l_-}$; $w_1 = \left[w_+^\top, b_+\right]^\top \in \mathbb{R}^{n+1}$, $w_2 = \left[w_-^\top, b_-\right]^\top \in \mathbb{R}^{n+1}$. Further, for simplification, we use A_j and B_i to represent $\left(1 + H_j^\top w_1\right)$ and $\left(1 - G_i^\top w_2\right)$, respectively.

In accordance with the optimality condition, we obtain the following:

$$\nabla P_1(w_1) = \left(GG^\top + C_1 I\right)w_1 + \hat{H}s_1 = 0, \tag{4.30}$$

$$\nabla P_2(w_2) = \left(HH^\top + C_3 I\right)w_2 - \hat{G}s_2 = 0, \tag{4.31}$$

where $\hat{G} = \left[C_4G_1,\ldots,C_4G_{l_+}\right] \in \mathbb{R}^{(n+1)\times l_+}$, $\hat{H} = \left[C_2H_1,\ldots,C_4H_{l_-}\right] \in \mathbb{R}^{(n+1)\times l_-}$. $s_1 = \left[s_{11},\ldots,s_{1l_-}\right]^\top \in \mathbb{R}^{l_-}$, $s_2 = \left[s_{21},\ldots,s_{2l_+}\right]^\top \in \mathbb{R}^{l_+}$; $s_{1j} = \frac{A_j(aA_j+2)\exp(aA_j)}{\{1+\lambda A_j^2\exp(aA_j)\}^2},\ j = 1,\ldots,l_-;\ s_{2i} = \frac{B_i(aB_i+2)\exp(aB_i)}{\{1+\lambda B_i^2\exp(aB_i)\}^2},\ i = 1,\ldots,l_+$.

Now, we use equations (4.30) and (4.31) to formulate iterative equations for optimization

problems (4.28) and (4.29) in the following manner:

$$w_1^{t+1} = -\left(GG^\top + C_1 I\right)^{-1} \hat{H} s_1^t, \tag{4.32}$$

$$w_2^{t+1} = \left(HH^\top + C_3 I\right)^{-1} \hat{G} s_2^t. \tag{4.33}$$

Here, t represents the number of iteration. The iterative procedure involves iterating through equations (4.32) and (4.33) until convergence is achieved. After obtaining the solutions, we can proceed to find the pair of hyperplanes (4.25).

To ascertain the class of a unseen sample $\widetilde{x} \in \mathbb{R}^n$, we use the following decision rule:

$$\text{Class of } \widetilde{x} = \begin{cases} +1, & \text{if } \frac{\left|w_+^\top \widetilde{x} + b_+\right|}{\|w_+\|} \leq \frac{\left|w_-^\top \widetilde{x} + b_-\right|}{\|w_-\|}, \\ -1, & \text{otherwise.} \end{cases} \tag{4.34}$$

4.2.5.2 Nonlinear Wave-TSVM

In the case of non-linearity, data points are separated linearly in a higher-dimensional feature space by utilizing the kernel trick to map them to a higher feature space. The objective of non-linear Wave-TSVM is to identify a pair of hypersurfaces in the following manner:

$$\mathcal{K}\left(x, X^\top\right) v_+ + b_+ = 0 \quad \text{and} \quad \mathcal{K}\left(x, X^\top\right) v_- + b_- = 0, \tag{4.35}$$

where $X = [X_+; X_-]^\top$ and $\mathcal{K}(\cdot, \cdot)$ is the kernel function. To determine the hypersurfaces (4.35), we formulate the following optimization problems:

(Non-linear Wave-TSVM-1)

$$\min_{v_+,b_+,\zeta^-} \sum_{i=1}^{l_+} \frac{1}{2}\left(\mathcal{K}\left(x_i, X^\top\right) v_+ + b_+\right)^2 + \frac{1}{2}C_1\left(\|v_+\|_2^2 + b_+^2\right) + C_2 \sum_{j=1}^{l_-} \zeta_j^-,$$

$$\text{s.t.} \quad \zeta_j^- = \frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1+\mathcal{K}\left(x_j, X^\top\right) v_+ + b_+\right)^2 \exp\{a\left(1+\mathcal{K}\left(x_j, X^\top\right) v_+ + b_+\right)\}}\right),$$

$$j = 1, \ldots, l_-, \tag{4.36}$$

(Non-linear Wave-TSVM-2)

$$\min_{v_-,b_-,\zeta^+} \sum_{j=1}^{l_-} \frac{1}{2}\left(\mathcal{K}\left(x_j, X^\top\right) v_- + b_-\right)^2 + \frac{1}{2}C_3\left(\|v_-\|_2^2 + b_-^2\right) + C_4 \sum_{i=1}^{l_+} \zeta_i^+,$$

$$\text{s.t.} \quad \zeta_i^+ = \frac{1}{\lambda}\left(1 - \frac{1}{1+\lambda\left(1-\mathcal{K}\left(x_i, X^\top\right) v_- - b_-\right)^2 \exp\{a\left(1-\mathcal{K}\left(x_i, X^\top\right) v_- - b_-\right)\}}\right),$$

$$i = 1, \ldots, l_+. \tag{4.37}$$

The method for solving problems (4.36) and (4.37) is akin to the linear scenario. The iterative method corresponds to equations (4.36) and (4.37) can be derived as follows:

$$v_1^{t+1} = -\left(MM^\top + C_1 I\right)^{-1} \left(\sum_{j=1}^{l_-} C_2 N_j \frac{\left(1+N_j^\top v_1^t\right)\left(a\left(1+N_j^\top v_1^t\right)+2\right)\exp\left(a\left(1+N_j^\top v_1^t\right)\right)}{\{1+\lambda\left(1+N_j^\top v_1^t\right)^2 \exp\left(a\left(1+N_j^\top v_1^t\right)\right)\}^2}\right), \tag{4.38}$$

$$v_2^{t+1} = \left(NN^\top + C_3 I\right)^{-1} \left(\sum_{i=1}^{l_+} C_4 M_i \frac{\left(1-M_i^\top v_2^t\right)\left(a\left(1-M_i^\top v_2^t\right)+2\right)\exp\left(a\left(1-M_i^\top v_2^t\right)\right)}{\{1+\lambda\left(1-M_i^\top v_2^t\right)^2 \exp\left(a\left(1-M_i^\top v_2^t\right)\right)\}^2}\right). \tag{4.39}$$

Here $M = \left[\mathcal{K}\left(X_+, X^\top\right), e_1\right]^\top \in \mathbb{R}^{(l+1)\times l_+}$, $N = \left[\mathcal{K}\left(X_-, X^\top\right), e_2\right]^\top \in \mathbb{R}^{(l+1)\times l_-}$; M_i is the ith column of the matrix M, N_j is the jth column of the matrix N. $v_1 = \left[v_+^\top, b_+\right]^\top$, $v_2 = \left[v_-^\top, b_-\right]^\top$. Further, for simplification, we use E_j and F_i to represent $\left(1+N_j^\top v_1\right)$ and $\left(1-M_i^\top v_2\right)$, respectively.

It's important to highlight that equations (4.38) and (4.39) entail the intricate calculation of matrix inversion. Therefore, we utilized the Sherman-Morrison-Woodbury (SMW) theorem [86] to alleviate computational complexity. Subsequently, in equations (4.38) and (4.39), the inverse matrices are substituted with the following matrices:

$$Q_1 = \frac{1}{C_1}\left(I - M\left(C_1 I + M^\top M\right)^{-1} M^\top\right), \tag{4.40}$$

$$Q_2 = \frac{1}{C_3}\left(I - N\left(C_3 I + N^\top N\right)^{-1} N^\top\right). \tag{4.41}$$

Using the equations (4.40) and (4.41), the iterative approach can be derived in the following manner:

$$v_1^{t+1} = -\, Q_1 \hat{N} s_1^t, \tag{4.42}$$

$$v_2^{t+1} = Q_2 \hat{M} s_2^t, \tag{4.43}$$

where t denotes the number of iteration. Also, $\hat{N} = \left[C_2 N_1, \ldots, C_2 N_{l_-}\right] \in \mathbb{R}^{(l+1)\times l_-}$, and $\hat{M} = \left[C_4 M_1, \ldots, C_4 M_{l_+}\right] \in \mathbb{R}^{(l+1)\times l_+}$. Additionally, $s_1^t \in \mathbb{R}^{l_-}$, $s_{1j}^t = \frac{E_j^t\left(aE_j^t+2\right)\exp\left(aE_j^t\right)}{\{1+\lambda E_j^{t\,2}\exp\left(aE_j^t\right)\}^2}$, $j = 1, \ldots, l_-$; $s_2^t \in \mathbb{R}^{l_+}$, $s_{2i}^t = \frac{F_i^t\left(aF_i^t+2\right)\exp\left(aF_i^t\right)}{\{1+\lambda F_i^{t\,2}\exp\left(aF_i^t\right)\}^2}$, $i = 1, \ldots, l_+$. The iterative procedure can be established by iterating through equations (4.42) and (4.43) until convergence is achieved. Consequently, upon obtaining the solutions v_+, b_+ and v_-, b_-, we can then determine the positive and negative hypersurfaces generated by the kernel.

For a new sample $\widetilde{x} \in \mathbb{R}^n$, we use the following decision function:

$$\text{Class of } \widetilde{x} = \begin{cases} +1, & \text{if } \frac{\left|\mathcal{K}\left(\widetilde{x}, X^\top\right)v_+ + b_+\right|}{\sqrt{v_+^\top \mathcal{K}\left(X, X^\top\right)v_+}} \leq \frac{\left|\mathcal{K}\left(\widetilde{x}, X^\top\right)v_- + b_-\right|}{\sqrt{v_-^\top \mathcal{K}\left(X, X^\top\right)v_-}}, \\ -1, & \text{otherwise.} \end{cases} \tag{4.44}$$

The iterative algorithm structure for non-linear Wave-TSVM subproblem (4.36) is clearly described in Algorithm 4.2. The structure for subproblem (4.37) is similar to it.

Algorithm 4.2 Non-linear Wave-TSVM

Input:
Training dataset: $\{x_i, y_i\}_{i=1}^{l}$, $y_i \in \{-1, 1\}$;
The parameters: Convergence precision (η), maximum iteration number (T), parameter C_1 and C_2, wave loss parameters a and λ, iteration number $t = 0$;
Initialize: v_1^0;
Ensure: v_+, b_+;
$1: M = \left[\mathcal{K}\left(X_+, X^\top\right), e_1\right]^\top$, $N = \left[\mathcal{K}\left(X_-, X^\top\right), e_2\right]^\top$.
$2:$ **while** $t \leq T$ **do**
$3:$ **for** $j \leftarrow 1$ to l_- **do**
$4:$ $s_{1j}^t \leftarrow \frac{E_j^t\left(aE_j^t+2\right)\exp\left(aE_j^t\right)}{\{1+\lambda {E_j^t}^2 \exp\left(aE_j^t\right)\}^2}$.
$5:$ **end for**
$6:$ $v_1^{t+1} \leftarrow -Q_1\hat{N}s_1^t$
$7:$ **if** $\left\|v_1^{t+1} - v_1^t\right\| < \eta$ **then**
$8:$ break
$9:$ **else**
$10:$ $t \leftarrow t + 1$
$11:$ **end if**
$12:$ **end while**
$13: \left(v_+^\top, b_+\right)^\top = v_1^{t+1}$.

4.2.6 Computational Complexity

In this subsection, we discuss the computational complexity of the proposed Wave-SVM and Wave-TSVM. Let l and n denote the number of samples and features in training dataset, respectively, and k denote the size of the mini-batch. The l_+ and l_- denote the count of positive and negative samples, respectively. The computational complexity of Wave-SVM is dominated by the computation of the gradients, characterized by a complexity of $\mathcal{O}(k^2)$. Further, the process of computing the moving averages of the gradients' first and second moments, as well as performing bias correction for these averages, both require $\mathcal{O}(k)$ operations, as we have k model parameter to compute. Therefore, the overall computational complexity of the Wave-SVM can be considered as $\mathcal{O}(T(k^2 + k))$, where T represents the maximum number of iterations. This encapsulates the fundamental operations involved in training the Wave-SVM and underscores its scalability concerning the number of iterations

and features. The computational complexity of Wave-TSVM primarily arises from the computation of matrix inversion. In the linear case, the algorithm requires solving the inverse of a $(n+1) \times (n+1)$ matrix, which results in a time complexity of $\mathcal{O}((n+1)^3)$. For the non-linear case, the algorithm needs to compute the inverse of two matrices: one of size $l_+ \times l_+$ and the other of size $l_- \times l_-$, with computational complexities of $\mathcal{O}(l_+^3)$ and $\mathcal{O}(l_-^3)$, respectively. Hence, for non-linear Wave-TSVM, the time complexity is $\mathcal{O}(T(l_+^3 + l_-^3))$. It is evident that non-linear Wave-TSVM is not well-suited for large-scale datasets due to its cubic computational complexity growth in relation to the sample size.

4.3 Numerical Experiments

In this section, we assess the proposed Wave-SVM and Wave-TSVM against the baseline models, including C-SVM [6], Pin-SVM [27], LINEX-SVM [31], FP-SVM [99], TSVM [10], Pin-GTSVM [88], SLTSVM [89], and IF-RVFL [101]. For experiments, we downloaded 42 binary datasets of diverse domains from the UCI [80] and KEEL [79] repositories. Table 4.2 summarizes the UCI and KEEL datasets used in the experiments. Additionally, we assess the models on the Alzheimer's disease (AD) dataset, available on the Alzheimer's Disease Neuroimaging Initiative (ADNI) $(adni.loni.usc.edu)$.

4.3.1 Experimental Setup and Parameter Selection

All the experiments are implemented using MATLAB R2023a on Windows 10 running on a PC with configuration Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 3408 Mhz, 4 Core(s), 8 Logical Processor(s) with 16 GB of RAM. For each SVM type model (C-SVM [6], Pin-SVM [27], LINEX-SVM [31], FP-SVM [99], and proposed Wave-SVM), the regularization parameter C is selected from the set $\{10^i \,|\, i = -6, -5, \ldots, 5, 6\}$. For Pin-SVM, the hyperparameter τ is chosen from the set $\{0, 0.1, \ldots, 0.9, 1\}$. For LINEX-

SVM and FP-SVM, the loss hyperparameters are selected the same as in [31] and [99], respectively. For the proposed Wave-SVM, loss hyperparameters λ and a are selected from the ranges $[0.1 : 0.2 : 2]$ and $[-2 : 0.1 : 5]$, respectively. For the non-linear case, Gaussian kernel function $\mathcal{K}(x_j, x_k) = \exp\left(-\|x_j - x_k\|^2 / \sigma^2\right)$ (where σ is the kernel parameter) is considered. The Gaussian kernel parameter σ for each SVM type model is selected from the set $\{10^i \mid i = -6, -5, \ldots, 5, 6\}$. The parameters for the Adam algorithm are experimentally set as: (i) initial weight $w_0 = 0.01$, (ii) initial first moment $m_0 = 0.01$, (iii) initial second moment $v_0 = 0.01$, (iv) initial learning rate α is selected from the set $\{0.0001, 0.001, 0.01\}$, (v) exponential decay rate of first order $\beta_1 = 0.9$, (vi) exponential decay rate of second order $\beta_2 = 0.999$, (vii) error tolerance $\eta = 10^{-5}$, (viii) division constant $\epsilon = 10^{-8}$, (ix) mini-batch size $k = 2^5$, (x) maximum iteration number $T = 1000$.

For each TSVM type model (TSVM [10], Pin-GTSVM [88], SLTSVM [89], and proposed Wave-TSVM), the regularization parameters (C_1 and C_3) and kernel parameter (σ) are selected from the set $\{10^i \mid i = -6, -4, \ldots, 4, 6\}$. For Pin-GTSVM, the hyperparameter τ is selected the same as in [88]. For SLTSVM and the proposed Wave-TSVM, the structural risk minimization parameters C_2 and C_4 are selected from the set $\{10^i \mid i = -6, -4, \ldots, 4, 6\}$. For SLTSVM and the proposed Wave-TSVM, we set $C_1 = C_3$ and $C_2 = C_4$. For iterative algorithm, to solve SLTSVM and the proposed Wave-TSVM, we set $\eta = 10^{-5}$ and $T = 50$. To suppress the tuning time for the proposed Wave-TSVM on UCI and KEEL datasets, the loss hyperparameter λ is fixed to 1, and a is selected from the range $[-2 : 0.5 : 2]$. For IF-RVFL [101], the regularization parameter (C) and kernel parameter (σ) are selected from the set $\{10^i \mid i = -6, -4, \ldots, 4, 6\}$, and the number of hidden nodes (N_d) is selected from the range $[3 : 20 : 203]$ [102].

The performance of the models significantly depends on the selection of parameters [103]. In order to adjust them, we tune the parameters based on k-fold (k$= 4$) cross-

validation and grid search. The benefit behind k-fold cross-validation is that it ensures that every sample will eventually be included in both the training and testing sets. More specifically, the dataset is divided into four non-overlapping subsets at random, one of which is set aside as a test set and the other three as train set. This process is conducted four times, and at every setup, a different test set is chosen while the other three sets are used as training, and the best of the four testing results is used as the performance measure for each model. For all the experiments, we normalized the data in the range of $[0, 1]$. The folds, preprocessing, parameter-search strategy, and selection criterion are kept the same for all baseline and proposed models, enabling a controlled comparison under one common setup. However, using the held-out folds both to select the best configuration or fold result and to report performance does not provide an independent estimate of generalization. Such reuse can introduce optimistic model-selection bias [78]. The reported differences should therefore be interpreted within this common four-fold protocol; a nested cross-validation design or a distinct validation split followed by an untouched test set would provide stronger confirmation on new data.

Table 4.2: Description of UCI and KEEL datasets employed in the experiments.

Dataset	Number of samples	Number of features	I.R.
breast_cancer	286	9	2.36
breast_cancer_wisc	699	9	1.9
breast_cancer_wisc_diag	569	30	1.68
congressional_voting	435	16	1.59
conn_bench_sonar_mines_rocks	208	60	1.14
credit_approval	690	15	1.25
cylinder_bands	512	35	1.56
echocardiogram	131	10	2.05
haberman_survival	306	3	2.78
hepatitis	155	19	3.84
horse_colic	368	25	1.71
monks_1	556	6	1
oocytes_trisopterus_nucleus_2f	1022	41	1.37
pima	768	8	1.86
pittsburg_bridges_T_OR_D	102	7	6.28
spect	265	22	1.41
statlog_heart	270	13	1.25
vertebral_column_2clases	310	6	2.1
abalone9-18	731	7	16.39
aus	690	14	1.25
cmc	1473	9	1.35
ecoli0137vs26	311	7	4.76
ecoli01vs5	240	7	11
ecoli3	336	7	8.6
led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	10.98
monk1	556	6	1.04
monk3	556	6	1.01
shuttle-6_vs_2-3	230	9	21.98
shuttle-c0-vs-c4	1829	9	13.87
sonar	208	60	1.14
transfusion	748	4	3.2
votes	435	16	1.59
vowel	988	10	9.99
yeast-0-2-5-6_vs_3-7-8-9	1004	8	9.14
yeast-0-3-5-9_vs_7-8	506	8	9.12
yeast1vs7	459	8	14.3
yeast2vs8	483	8	23.15
ringnorm	7400	20	1.02
twonorm	7400	20	1
mushroom	8124	21	1.07
musk_2	6598	166	5.49
EEG Eye State	14980	14	1.23

To assess the efficacy of the proposed Wave-SVM and Wave-TSVM against baseline models, we employed the accuracy metric, defined as follows:

$$\text{Accuracy} = \frac{\text{T.P.+T.N.}}{\text{T.P.+T.N.+F.P.+F.N.}} \times 100. \tag{4.45}$$

Here, T.P., T.N., F.P., and F.N. represent the true positive, true negative, false positive, and false negative, respectively. Further, we conducted a comparative analysis of the training time between the proposed Wave-SVM and the baseline models. The reported times only include the time taken for training the models with the optimized hyperparameters.

4.3.2 Evaluation on UCI and KEEL Datasets

For a fair comparison of the proposed Wave-SVM and Wave-TSVM with baseline models, we compare Wave-SVM with C-SVM, Pin-SVM, LINEX-SVM, and FP-SVM, and Wave-TSVM with TSVM, Pin-GTSVM, and SLTSVM. In essence, we compared Wave-SVM and Wave-TSVM with SVM type models and TSVM type models, respectively. Additionally, we compare Wave-TSVM with IF-RVFL.

First, we discuss the experimental outcomes of Wave-SVM. For linear case, the average classification accuracy and training time of the proposed Wave-SVM and the baseline models (C-SVM, Pin-SVM, LINEX-SVM, and FP-SVM) are presented in Table 4.3. The average accuracy of the existing C-SVM, Pin-SVM, LINEX-SVM, and FP-SVM are 82.21%, 89.1%, 77.41%, and 89.73% respectively, whereas, the average accuracy of the proposed Wave-SVM is 91.15%, surpassing the baseline models.

Table 4.3: Accuracy and training time for linear Wave-SVM against baseline models on UCI and KEEL datasets.

Dataset	C-SVM [6]		Pin-SVM [27]		LINEX-SVM [31]		FP-SVM [99]		Wave-SVM†	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
breast_cancer	100	0.004	100	0.543	94.44	0.0059	100	0.3012	100	0.0059
breast_cancer_wisc	95.43	0.0351	100	0.1989	99.43	0.003	99.43	0.0759	99.43	0.003
breast_cancer_wisc_diag	96.5	0.1831	99.3	0.1108	93.66	0.0044	98.59	1.316	99.3	0.0044
congressional_voting	64.22	0.011	64.22	0.0702	58.72	0.0032	64.22	0.2424	66.97	0.0032
conn_bench_sonar_mines_rocks	61.54	0.018	100	0.0168	90.38	0.0075	100	0.0717	100	0.0075
credit_approval	69.94	0.0407	99.42	0.196	63.01	0.0013	99.42	1.8022	99.42	0.0013
cylinder_bands	67.97	0.0127	73.44	0.0735	72.66	0.0043	74.22	0.7981	75.78	0.0043
echocardiogram	81.82	0.002	93.75	0.0094	93.75	0.0035	93.75	0.7435	96.97	0.0035
haberman_survival	80.52	0.0081	80.52	0.0429	83.12	0.005	81.82	12.139	84.42	0.005
hepatitis	94.87	0.0038	94.87	0.0103	69.23	0.0057	94.87	0.1911	97.44	0.0057
horse_colic	82.61	0.0087	92.39	0.0423	51.09	0.0021	88.04	0.4263	90.22	0.0021
monks_1	50.36	0.0174	63.31	0.1267	67.63	0.0013	74.1	1.0576	87.05	0.0013
oocytes_trisopterus_nucleus_2f	74.56	0.3941	76.75	0.5174	67.98	0.0014	76.75	1.5174	78.95	0.0014
pima	75.52	0.0358	81.25	0.2001	70.31	0.0011	79.69	2.2016	82.29	0.0011
pittsburg_bridges_T_OR_D	92.31	0.0019	96.15	0.0028	92.31	0.0009	96	0.0105	100	0.0009
spect	71.64	0.0053	77.27	0.0264	65.67	0.0017	78.79	0.2066	81.82	0.0017
statlog_heart	80.88	0.0056	89.55	0.0232	77.94	0.0015	88.06	0.021	92.54	0.0015
vertebral_column_2clases	100	0.219	100	0.0274	93.51	0.0009	100	0.1121	100	0.0009
abalone9-18	91.8	0.0955	100	0.2191	100	0.0015	100	0.5245	100	0.0129
aus	87.86	0.0731	91.28	0.0926	82.56	0.0022	90.7	1.7916	92.44	0.0025
cmc	10.3	0.3185	100	0.4872	100	0.0092	100	0.4161	100	0.0036
ecoli0137vs26	93.59	0.0243	97.44	0.0254	84.62	0.0017	93.59	0.3243	96.15	0.0014
ecoli01vs5	100	0.0122	100	0.0246	95	0.0022	100	0.4122	100	0.0021
ecoli3	91.67	0.0363	91.67	0.0422	91.67	0.0018	91.67	0.3363	92.86	0.0019
led7digit-0-2-4-5-6-7-8-9_vs_1	93.69	0.0183	97.3	0.0452	67.27	0.0024	98.2	0.415	96.4	0.0037
monk1	53.96	0.0179	53.96	0.0802	55.4	0.0019	56.12	0.6479	58.99	0.0036
monk3	51.8	0.0253	51.08	0.0677	56.12	0.002	55.4	13.2818	59.71	0.0022
shuttle-6_vs_2-3	100	0.0117	100	0.0195	82.76	0.0021	100	0.0081	100	0.0072
shuttle-c0-vs-c4	100	0.8325	100	1.1571	92.58	0.0019	100	0.8325	100	0.0026
sonar	78.85	0.0095	84.62	0.0154	61.54	0.0025	78.85	0.0095	86.54	0.0059
transfusion	59.36	0.0469	87.17	0.2267	40.64	0.0022	87.7	10.1106	87.17	0.0012
votes	96.33	0.0245	97.25	0.032	77.06	0.0022	97.25	0.0858	97.25	0.0026
vowel	95.55	0.1324	96.76	0.266	91.09	0.0019	95.55	0.1324	96.76	0.0026
yeast-0-2-5-6_vs_3-7-8-9	94.02	0.1471	94.02	0.2317	93.23	0.0028	96.41	21.9554	94.42	0.002
yeast-0-3-5-9_vs_7-8	92.13	0.0298	95.24	0.0416	91.34	0.0019	96.83	6.0563	95.24	0.0021
yeast1vs7	94.78	0.0428	94.78	0.0415	94.74	0.0031	94.78	1.428	94.78	0.0033
yeast2vs8	98.35	0.1581	98.35	0.0513	47.11	0.0039	99.17	1.3929	99.17	0.0056
ringnorm	77.51	18.7724	78.05	22.3657	78.16	0.00147	*	*	79.09	0.0105
twonorm	98.1	13.1825	98.81	20.0629	34.59	0.0057	*	*	98.92	0.0089
mushroom	94.69	17.1168	95.27	66.9571	79.02	0.0027	*	*	99.21	0.0044
musk_2	99.15	12.3079	100	17.5185	93.21	0.0207	*	*	98.24	0.0513
EEG Eye State	58.83	129.8739	56.98	164.8006	56.82	0.0051	*	*	72.55	0.0089
Avg. Acc. and Avg. time	82.21	4.6266	89.1	7.074	77.41	**0.0033**	89.73	2.254	**91.15**	0.0049
Avg. Rank	3.73		2.52		4.29		2.49		**1.68**	

† represents the proposed model.

* denotes that we stopped the implementation after 36 hours.

Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.

The average training time, expressed in seconds, of C-SVM, Pin-SVM, LINEX-SVM, FP-SVM, and the proposed Wave-SVM are 4.6266s, 7.074s, 0.0033s, 2.254, and 0.0049s, respectively. This observation strongly underscores the substantial superiority of the proposed Wave-SVM over C-SVM and Pin-SVM. However, its training time is marginally extended compared to LINEX-SVM. Nonetheless, it still demonstrates commendable performance in terms of training time. For the non-linear case, the average accuracy and training time of the proposed Wave-SVM and baseline models are presented in Table 4.4. The average accuracy of C-SVM, Pin-SVM, LINEX-SVM, FP-SVM, and the proposed Wave-SVM are 89.46%, 89.97%, 86.63%, 90.18%, and 91.92%, respectively. Evidently, the proposed Wave-SVM achieves the highest classification accuracy in comparison with the baseline models. The average training time of C-SVM, Pin-SVM, LINEX-SVM, FP-SVM, and the proposed Wave-SVM are 7.57s, 8.64s, 0.0137s, 1.39s, and 0.0219s, respectively.

Table 4.4: Accuracy and training time for non-linear Wave-SVM against baseline models using Gaussian kernel function on UCI and KEEL datasets.

Dataset	C-SVM [6]		Pin-SVM [27]		LINEX-SVM [31]		FP-SVM [99]		Wave-SVM†	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
breast_cancer	100	0.2574	100	0.4802	100	0.0108	100	0.137	100	0.0069
breast_cancer_wisc	100	0.0644	100	0.1535	98.28	0.0038	100	25.4471	100	0.0031
breast_cancer_wisc_diag	97.89	0.0633	97.89	0.0664	96.48	0.0049	98.59	0.0246	98.59	0.0082
congressional_voting	65.14	0.0281	65.14	0.033	63.3	0.0044	65.74	0.0134	67.89	0.0078
conn_bench_sonar_mines_rocks	100	0.0206	100	0.0173	100	0.0053	100	0.0052	100	0.0033
credit_approval	95.38	0.138	96.53	0.1659	90.75	0.005	95.38	0.0404	95.95	0.0108
cylinder_bands	71.09	0.0305	71.09	0.0478	71.09	0.0046	74.22	0.0315	76.56	0.015
echocardiogram	84.38	0.0036	87.5	0.006	84.38	0.0034	93.75	0.0019	96.97	0.0072
haberman_survival	80.52	0.018	81.82	0.0475	79.22	0.0054	80.52	0.0077	85.71	0.0045
hepatitis	92.31	0.0142	92.31	0.0052	94.87	0.0041	92.31	0.0144	97.44	0.007
horse_colic	76.09	0.0201	83.7	0.0314	75	0.0048	85.87	2.0631	85.87	0.0039
monks_1	72.66	0.0791	74.1	0.0642	68.35	0.0046	79.14	0.0545	85.61	0.0066
oocytes_trisopterus_nucleus_2f	72.63	0.2335	72.63	0.2542	67.54	0.0051	72.63	0.3335	75	0.0062
pima	79.69	0.0822	80.21	0.1601	73.44	0.0045	79.69	0.0632	80.73	0.0083
pittsburg_bridges_T_OR_D	96	0.0047	96	0.005	96	0.0046	96.15	0.4534	100	0.0016
spect	74.24	0.0162	75.76	0.0167	72.73	0.0042	77.27	0.0049	86.36	0.0062
statlog_heart	86.57	0.021	86.57	0.0164	83.58	0.0041	88.06	0.0074	89.71	0.0043
vertebral_column_2clases	100	0.0139	100	0.0143	100	0.0045	100	0.0399	100	0.0071
abalone9-18	100	0.0747	100	0.1922	100	0.0054	100	0.3086	100	0.0076
aus	84.97	0.1197	89.53	0.14	87.21	0.0049	88.37	19.5134	90.12	0.0085
cmc	99.73	2.3671	100	1.5839	87.5	0.0099	100	0.1654	100	0.0118
ecoli0137vs26	98.7	0.0172	98.72	0.0199	84.62	0.0044	98.72	0.0199	98.72	0.0104
ecoli01vs5	100	0.0097	100	0.0173	95	0.004	100	0.0173	100	0.0083
ecoli3	95.24	0.0163	96.43	0.0447	91.67	0.0043	96.43	0.0447	100	0.0088
led7digit-0-2-4-5-6-7-8-9_vs_1	98.2	0.0435	98.2	0.0311	96.4	0.0046	98.2	0.024	98.2	0.0073
monk1	54.68	0.0892	54.68	0.1369	58.99	0.0048	56.12	0.0886	65.47	0.0073
monk3	52.52	0.0785	51.08	0.0678	57.55	0.0045	57.55	1.2189	64.03	0.0095
shuttle-6_vs_2-3	100	0.0132	100	0.0099	100	0.0047	100	0.0231	100	0.0095
shuttle-c0-vs-c4	99.78	0.5597	100	0.5717	99.78	0.0053	100	0.2061	99.78	0.0042
sonar	92.31	0.0078	92.31	0.0215	75	0.0049	92.31	0.0043	88.46	0.0078
transfusion	87.17	0.0562	87.17	0.1295	87.17	0.0038	87.7	0.034	87.17	0.0063
votes	96.33	0.0285	96.33	0.0274	94.5	0.0053	96.17	0.1345	96.33	0.0067
vowel	97.98	0.3369	97.98	0.3629	95.55	0.0135	97.98	0.9215	100	0.007
yeast-0-2-5-6_vs_3-7-8-9	95.22	0.2745	95.22	0.4243	93.23	0.0047	96.02	0.0356	97.21	0.0065
yeast-0-3-5-9_vs_7-8	95.24	0.0448	95.24	0.094	92.86	0.004	95.24	0.1141	97.62	0.0036
yeast1vs7	97.39	0.0557	97.39	0.0553	94.78	0.0052	97.39	0.0553	98.26	0.0038
yeast2vs8	98.35	0.0187	98.35	0.0469	98.35	0.0058	99.17	0.0141	100	0.0145
ringnorm	100	12.801	100	11.9823	97.44	0.0891	*	*	79.51	0.1245
twonorm	100	4.9025	100	6.4691	100	0.189	*	*	98.65	0.303
mushroom	97.89	29.4162	98.05	15.2824	60.27	0.0284	*	*	100	0.0614
musk_2	98.54	15.9498	98.54	5.851	95.95	0.0409	*	*	100	0.0617
EEG Eye State	72.52	249.8753	72.52	317.6164	79.87	0.0344	*	*	78.8	0.1007
Avg. Acc. and Avg. time	89.46	7.57	89.97	8.64	86.63	**0.0137**	90.18	1.39	**91.92**	0.0219
Avg. Rank	3.37		2.85		4.16		2.53		**1.8**	

† represents the proposed model.
Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.

The outcomes manifest that the proposed Wave-SVM exhibits remarkable efficiency. Its training time is significantly less than C-SVM and Pin-SVM, indicating its enhanced computational efficiency. However, the proposed Wave-SVM takes a slightly longer time to train than the LINEX-SVM, although the difference in training time is minor. It is noteworthy that the comparative evaluation of accuracy and training time shows the overall performance of the models. Thus, the proposed Wave-SVM's ability to strike a balance between classification accuracy and training time makes it a superior model in comparison to the baseline models. To assess the TSVM type models, given the substantial computational complexity of the proposed Wave-TSVM due to the matrix inversion, we excluded datasets with either over 5000 samples or more than 25 features. This exclusion resulted in a remaining set of 32 datasets. The average experimental outcomes for the non-linear proposed Wave-TSVM and baseline models, including TSVM, Pin-GTSVM, SLTSVM, and IF-RVFL, across these 32 datasets are presented in Table 4.5. The average classification accuracies of the existing TSVM, Pin-GTSVM, IF-RVFL, and SLTSVM are 86.49%, 83.24%, 83.8%, and 85.98%, respectively. In contrast, the average accuracy of the proposed Wave-TSVM is 87.91%, surpassing that of baseline models. These findings strongly emphasize the significant superiority of the proposed Wave-TSVM over the baseline models.

Table 4.5: Accuracy for non-linear Wave-TSVM against baseline models on UCI and KEEL datasets.

Dataset	TSVM [10]	Pin-GTSVM [88]	IF-RVFL [101]	SLTSVM [89]	Wave-TSVM†
breast_cancer	100	100	100	98.59	100
breast_cancer_wisc	88.01	88.68	83.57	78.16	87.93
congressional_voting	63.3	64.04	65.14	65.14	66.06
credit_approval	91.84	90.18	90.92	83.82	91.91
echocardiogram	87.2	80.74	85.63	87.5	87.5
haberman_survival	80.52	67.97	80.52	79.22	81.82
hepatitis	82.31	84.29	76.67	94.87	97.44
horse_colic	75.04	83.48	77.61	73.91	79.35
monks_1	73.38	78.48	74.82	89.93	87.77
pima	77.69	75.4	70.1	71.88	72.4
pittsburg_bridges_T_OR_D	96.15	95.83	93.08	96	96.15
spect	75.76	77.5	63.64	68.18	73.13
statlog_heart	65.07	66.47	66.72	67.65	67.65
vertebral_column_2clases	100	100	100	100	100
abalone9-18	100	88.01	90.48	100	100
aus	76.95	78.37	75.93	78.03	77.91
cmc	100	67.08	100	93.22	100
ecoli0137vs26	93.12	91.44	91.82	92.31	94.87
ecoli01vs5	94.35	90.36	86.67	95	95
ecoli3	91.4	89.4	89.29	92.86	92.86
led7digit-0-2-4-5-6-7-8-9_vs_1	95.3	93.31	89.37	93.69	96.4
monk1	55.4	52.76	61.15	56.83	56.83
monk3	57.55	56.58	56.83	56.83	55.4
shuttle-6_vs_2-3	100	100	98.28	100	100
shuttle-c0-vs-c4	95.76	95.76	94.09	97.16	96.72
transfusion	87.17	83.87	87.17	87.17	87.7
votes	90.17	90.48	78.53	79.82	92.66
vowel	92.19	96.43	95.08	91.5	95.14
yeast-0-2-5-6_vs_3-7-8-9	94.42	84.6	94.02	93.63	93.23
yeast-0-3-5-9_vs_7-8	95.24	81.62	88.9	95.24	95.24
yeast1vs7	93.1	82.03	85.65	94.78	94.78
yeast2vs8	99.17	88.66	89.92	98.35	99.17
Avg. Acc.	86.49	83.24	83.8	85.98	**87.91**
Avg. Rank	2.78	3.55	3.75	2.92	**2**

† represents the proposed model.
Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.

The statistical analysis is presented after the main empirical results so that the dataset-wise performance trends can be interpreted before formal significance testing.

Table 4.6: Performance comparison of the proposed Wave-SVM compare baseline models on benchmark UCI and KEEL datasets with Gaussian noise.

Model		C-SVM [6]	Pin-SVM [27]	LINEX-SVM [31]	FP-SVM [99]	Wave-SVM†
Dataset	Noise	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
breast_cancer_wisc	5%	93.68	97.7	100	100	100
	10%	94.25	97.7	100	100	100
	20%	87.36	97.7	100	100	100
	30%	85.14	98.85	100	100	100
Avg. Acc.		90.11	<u>97.99</u>	**100**	**100**	**100**
credit_approval	5%	85.55	84.88	91.33	94.8	97.11
	10%	86.13	85.47	89.02	95.38	96.53
	20%	84.97	86.13	89.02	95.38	92.49
	30%	84.39	84.97	89.02	94.8	93.64
Avg. Acc.		85.26	85.36	89.6	**95.09**	<u>94.94</u>
horse_colic	5%	70.65	76.09	81.52	85.87	86.96
	10%	70.65	73.91	82.61	84.78	86.96
	20%	69.57	71.74	80.43	88.04	86.96
	30%	68.48	69.57	80.43	86.96	84.78
Avg. Acc.		69.84	72.83	<u>81.25</u>	**86.41**	**86.41**
led7digit-0-2-4-5-6-7-8-9_vs_1	5%	96.4	96.4	96.4	97.3	100
	10%	96.4	96.4	93.69	98.2	98.2
	20%	96.4	96.4	95.45	97.3	98.2
	30%	96.36	96.4	94.59	97.3	99.1
Avg. Acc.		96.39	96.4	95.03	<u>97.52</u>	**98.88**
monk1	5%	56.12	53.96	56.83	59.71	65.47
	10%	54.68	57.55	53.96	58.27	66.19
	20%	55.4	56.83	56.12	58.27	64.03
	30%	58.27	56.83	56.83	61.87	64.75
Avg. Acc.		56.12	56.29	55.93	<u>59.53</u>	**65.11**
yeast2vs8	5%	98.35	98.35	98.35	99.17	99.17
	10%	98.35	98.35	98.35	99.17	100
	20%	98.35	98.35	98.35	99.17	100
	30%	98.35	98.35	98.35	99.17	100
Avg. Acc.		98.35	98.35	98.35	<u>99.17</u>	**99.79**

† represents the proposed model.
Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.
The boldface and underline indicate the best and second-best models, respectively, in terms of average accuracy.

4.3.3 Evaluation on UCI and KEEL Datasets with Gaussian Noise

While the UCI and KEEL datasets used in our evaluation reflect real-world circumstances, it is essential to acknowledge that outliers or noise can arise due to various factors. To showcase the effectiveness of the proposed Wave-SVM and Wave-TSVM, even in adverse conditions, we deliberately added feature noise to selected datasets. We selected 6

diverse datasets for our comparative analysis, namely breast_cancer_wisc, credit_approval, horse_colic, led7digit-0-2-4-5-6-7-8-9_vs_1, monk1, and yeast2vs8. To ensure impartiality in evaluating the models, we selected 2 datasets where the proposed Wave-SVM achieves the highest performance compared to baseline models in the corresponding clean-data experiments. Further, we selected 2 datasets where the proposed Wave-SVM does not achieve the highest performance and 2 datasets where the proposed Wave-SVM ties with an existing model. To carry out a comprehensive evaluation, we added Gaussian noise at varying levels of 5%, 10%, 20%, and 30% to corrupt the features of these datasets. The accuracy values of Wave-SVM against baseline models for the selected datasets with 5%, 10%, 20%, and 30% level of noise are presented in Table 4.6. Out of 6 diverse datasets, the proposed Wave-SVM outperforms the baseline models on 5 datasets and attains second position on 1 dataset, as per the average accuracies at different levels of noise. This outcome unequivocally underscores the significance of the proposed Wave-SVM as a robust model. Similarly, the accuracy values of Wave-TSVM against baseline models for the selected datasets with 5%, 10%, 20%, and 30% noise levels are outlined in Table 4.7. Out of the 6 diverse datasets, the proposed Wave-TSVM surpasses the baseline algorithms on 4 datasets, achieves the second position on 1 dataset, and secures the third position on 1 dataset, based on the average accuracies at different noise levels. This result demonstrates the significance and robustness of Wave-TSVM. By subjecting the models to rigorous conditions, we demonstrate the exceptional performance and superiority of the proposed Wave-SVM and Wave-TSVM, especially in adverse scenarios.

Table 4.7: Performance comparison of the proposed Wave-TSVM compare baseline models on benchmark UCI and KEEL datasets with Gaussian noise.

Model		TSVM [10]	Pin-GTSVM [88]	IF-RVFL [101]	SLTSVM [89]	Wave-TSVM†
Dataset	Noise	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
breast_cancer_wisc	5%	88.01	88.68	83.57	78.16	87.36
	10%	88.01	88.68	83.57	78.16	85.63
	20%	86.9	87.37	83.57	78.16	83.91
	30%	84.37	85.5	83.57	78.16	80.46
Avg. Acc.		86.82	**87.56**	83.57	78.16	84.34
credit_approval	5%	91.84	90.18	80.92	80.92	91.33
	10%	89.53	87.57	80.92	80.81	89.02
	20%	86.37	87.57	80.92	79.65	88.44
	30%	85.95	84.08	80.92	73.41	86.13
Avg. Acc.		88.42	87.35	80.92	78.7	**88.73**
horse_colic	5%	75.04	82.25	77.61	75	77.17
	10%	75.04	79.9	77.61	71.74	84.78
	20%	73.22	79.9	77.61	70.65	76.09
	30%	69.04	68.86	77.61	72.83	76.09
Avg. Acc.		73.1	77.73	77.61	72.55	**78.53**
led7digit-0-2-4-5-6-7-8-9_vs_1	5%	95.3	93.81	89.37	95.5	98.2
	10%	95.3	87.09	89.37	95.5	95.5
	20%	93.5	87.09	89.37	93.69	95.45
	30%	91.4	84.5	89.37	93.69	94.59
Avg. Acc.		93.87	88.12	89.37	94.6	**95.93**
monk1	5%	55.71	52.15	61.15	61.15	58.99
	10%	56.99	52.15	61.15	60.43	60.43
	20%	58.15	53.98	61.15	57.55	60.43
	30%	59.71	51.18	61.15	61.15	60.43
Avg. Acc.		57.64	52.37	**61.15**	60.07	60.07
yeast2vs8	5%	99.17	88.55	89.92	97.52	98.35
	10%	98.35	88.55	89.92	98.35	99.17
	20%	97.69	88.55	89.92	98.35	98.35
	30%	96.04	88.1	89.92	97.52	98.35
Avg. Acc.		97.81	88.44	89.92	97.93	**98.55**

† represents the proposed model.

Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.

The boldface and underline indicate the best and second-best models, respectively, in terms of average accuracy.

4.3.4 Results Analysis on ADNI Dataset

Alzheimer's disease (AD) is a widely recognized neurodegenerative illness that progressively results in memory loss and cognitive impairments. AD development is irreversible and manifests as atrophy in the brain's inner regions. It is predicted that by the year 2050, one out of every 85 individuals will suffer from AD [104]. Multiple studies indicate that early detection and intervention can slow down the progression of AD. Therefore, to mitigate further growth, treatment should commence at the earliest possible stage. The ADNI project, initiated by Michael W. Weiner in 2003, aims to examine various neuroimaging techniques, including positron emission tomography (PET), magnetic resonance imaging (MRI), and other diagnostic tests for AD, particularly at the mild cognitive impairment (MCI) stage. AD data is publicly available on the ADNI repository, which is accessible at $adni.loni.usc.edu$. The pipeline for feature extraction opted in this chapter is the same as followed in [95]. The dataset contains three cases: control normal (CN) versus AD, CN versus MCI, and MCI versus AD.

The accuracy of the proposed Wave-SVM, as well as baseline models for AD diagnosis, are presented in Table 4.8. We examine that with an average accuracy of 79.11%, the proposed Wave-SVM is the best model. The average accuracies of baseline algorithms, C-SVM, Pin-SVM, and LINEX-SVM are 70.55%, 78.16%, and 75.99%, respectively. With an accuracy of 88.46% and 79.45%, the proposed Wave-SVM is the most accurate classifier for CN versus AD and MCI versus AD cases. For CN versus MCI case, Pin-SVM came out on top followed by the proposed Wave-SVM. Overall, the proposed Wave-SVM has emerged as the best classifier in comparison to the baseline models for AD diagnosis. The Wave-based models developed in this chapter primarily address robustness to noisy observations and outliers and are not specifically formulated for class-imbalance-aware learning. Accordingly, the ADNI experiments use accuracy as the principal predictive measure, with

identical data partitions, preprocessing, model-selection procedures, and evaluation settings applied to all baseline and proposed models. In contrast, Chapter 6 explicitly develops an imbalance-aware weighting mechanism and therefore reports F-measure together with accuracy.

Sensitivity, specificity, and F-measure would provide a more clinically informative evaluation of the three ADNI classification tasks. However, the retained experimental outputs do not contain the fold-level predictions or confusion matrices required to compute these measures reliably from the reported aggregate accuracies. A valid retrospective comparison would require rerunning every baseline and proposed model using the same data partitions, preprocessing, and model-selection procedure. Table 4.8 is therefore retained as an accuracy-based comparison, and its conclusions are restricted to comparative predictive accuracy under the stated protocol. The results should not be interpreted as a complete assessment of diagnostic sensitivity, specificity, or minority-class performance. Future evaluations should preserve fold-level outputs and report class-sensitive measures together with appropriate uncertainty estimates.

Table 4.8: Results of the proposed non-linear Wave-SVM against baseline models on ADNI dataset.

Model	C-SVM [6]		Pin-SVM [27]		LINEX-SVM [31]		FP-SVM [99]		Wave-SVM†	
Dataset	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
(# of samples, # of features)	(C, σ)		(τ, C, σ)		(a, C, σ)		(C, σ, τ_1, τ_2)		(λ, a, C, σ)	
CN versus AD	82.26	0.0206	84.61	0.006	86.54	0.0011	84.62	0.0668	88.46	0.0019
(415 ,91)	1,1		0,0.1,0.001		-1,0.001,1		0.000001, 0.000001, 0.5, 0		0.1,-0.6,1000,10	
CN versus MCI	63.1	0.0247	74.52	0.0119	68.15	0.0018	75.15	0.0241	69.43	0.0027
(626, 91)	10,1		0.2,0.1,0.001		-1,1000,10		0.0001, 0.01, 0, 0.4		0.7,3.2,0.001,10	
MCI versus AD	66.28	0.0176	75.34	0.0502	73.29	0.0014	73.97	0.0444	79.45	0.0024
(585, 91)	1,1		0,10,0.01		-1,0.001,1		1000000, 0.000001, 0, 0.8		0.1,0.3,100,10	
Avg. Acc. and Avg. Time	70.55	0.021	78.16	0.0227	75.99	**0.0014**	77.91	0.0451	**79.11**	0.0023

† represents the proposed model.
Here, Avg. and Acc. are acronyms used for average and accuracy, respectively.

4.4 Statistical Analysis of Wave-SVM and Wave-TSVM

To further support the efficacy of the proposed Wave-SVM and Wave-TSVM, we conducted a comprehensive statistical analysis of the models, for which we followed four tests:

ranking scheme, Friedman test, Nemenyi post hoc test, and Win-Tie-Loss sign test [25].

Ranking scheme: To conduct a more comprehensive evaluation of the Wave-SVM, we employed a ranking scheme. Relying solely on average accuracy as a single metric could be problematic, as exceptional performance on specific datasets might mask inadequate performance on others. To mitigate this concern, it becomes essential to individually rank each model with respect to each dataset, allowing for a comprehensive assessment of their respective capabilities. In the ranking scheme [25], the model with the poorest performance on a dataset is assigned a higher rank, while the model achieving the best performance is assigned a lower rank. Consider p models are being assessed using D datasets, and R_e^f denotes the rank of the eth of p models on the fth of D datasets. The eth model's mean rank is obtained by the following formula:

$$R_e = \frac{1}{D} \sum_f R_e^f. \tag{4.46}$$

For SVM type models, the mean rank of the baseline models C-SVM, Pin-SVM, LINEX-SVM, and FP-SVM are 3.37, 2.85, 4.16, and 2.53, respectively. Whereas, the mean rank of the proposed Wave-SVM is 1.8, the most favorable position when compared to the baseline models. For TSVM type models, the mean rank of the baseline models including, TSVM, Pin-GTSVM, IF-RVFL, and SLTSVM are 2.78, 3.55, 3.75, and 2.92, respectively. In contrast, the mean rank of the proposed Wave-TSVM is 2, the lowest in comparison to baseline models (see Table 4.5). Since the lower rank denotes heightened accomplishment, the proposed Wave-SVM and Wave-TSVM securely occupy the foremost positions in comparison to baseline models.

Friedman test: The Friedman test [24] is utilized to statistically assess the significance of various models. In this test, each individual model is ranked independently for each dataset. The highest-performing model receives a rank of 1, followed by the second-best

model with a rank of 2, and so forth. The null hypothesis posits that all models are essentially identical, implying that the average rank of each model is equivalent. The Friedman statistic adheres to the chi-squared χ_F^2 distribution with degrees of freedom (d.f.) equal to $p-1$, where p represents the number of models. This statistic is computed as follows:

$$\chi_F^2 = \frac{12D}{p(p+1)} \left[\sum_e R_e^2 - \frac{p(p+1)^2}{4} \right]. \tag{4.47}$$

Here, D signifies the total number of datasets, and R_e denotes the mean rank of the eth model out of the p models. However, the Friedman statistic is overly cautious in nature. To address this, Iman and Davenport [62] introduced a more robust statistic:

$$F_F = \frac{(D-1)\chi_F^2}{D(p-1) - \chi_F^2}, \tag{4.48}$$

which follows F distribution with $((p-1),(p-1)(D-1))$ d.f.. For the proposed Wave-SVM and baseline models, we have $p = 5$ and $D = 42$. Thus, we obtain $\chi_F^2 = 23.95$ and $F_F = 6.82$. Also, F_F is distributed with $(4, 164)$ d.f.. From the statistical F-distribution table, at 5% level of significance, the value of $F_F(4, 164) = 2.43$. Since, $F_F > 2.43$, thus we reject the null hypothesis. For the proposed Wave-TSVM and baseline models, we have $p = 5$ and $D = 32$. Thus, we obtain $\chi_F^2 = 24.58$ and $F_F = 7.36$. Also, F_F is distributed with $(4, 124)$ d.f.. From the statistical F-distribution table, at 5% level of significance, the value of $F_F(4, 124) = 2.45$. Since, $F_F > 2.45$, thus we reject the null hypothesis. Hence, substantial differences exist among the models.

Nemenyi post hoc test: In the Nemenyi post hoc test [25], a pairwise comparison is conducted among all models. The disparity in performance between the two models is considered significant if the corresponding average ranks display a noticeable difference surpassing a specific threshold value known as the critical difference ($C.D.$). When the dis-

crepancy in mean ranks between the two models exceeds the $C.D.$, the model with the lower mean rank is deemed statistically superior because rank 1 denotes the best performance. The computation of $C.D.$ follows the following formula:

$$C.D. = q_\alpha \sqrt{\frac{p(p+1)}{6D}}. \tag{4.49}$$

Here, q_α is derived from the studentized range statistic divided by $\sqrt{2}$. This value is referred to as the critical value for the two-tailed Nemenyi test. From distribution table, for $\alpha = 0.1$, the value of q_α is 2.29. Thus, after calculation, we obtain $C.D. = 0.85$ and 0.97 for SVM type models and TSVM type models, respectively. The mean rank difference of proposed Wave-SVM from C-SVM, Pin-SVM, LINEX-SVM, and FP-SVM are 1.57, 1.05, 2.36, and 0.73, respectively. Notably, Wave-SVM differs significantly from the baseline models C-SVM, Pin-SVM, and LINEX-SVM as the difference in the average rank is greater than $C.D.$, however, the rank difference of the proposed Wave-SVM with FP-SVM is less than $C.D.$. Similarly, the mean rank difference of proposed Wave-TSVM from TSVM, Pin-GTSVM, IF-RVFL, and SLTSVM are 0.78, 1.55, 1.75, and 0.92, respectively. Therefore, the proposed Wave-TSVM showcases significant disparity from the baseline models Pin-GTSVM and IF-RVFL, as the average rank difference surpasses the $C.D.$. Though, the rank difference of the proposed Wave-SVM with TSVM and SLTSVM is below the $C.D.$, the efficacy of the proposed Wave-TSVM over TSVM and SLTSVM is underscored by the average rank, average accuracy, and other statistical tests. These findings are supported by the data presented in Table 4.9 and 4.10, which clearly indicates the superior performance of the proposed Wave-SVM and Wave-TSVM in comparison to the baseline models.

Win-Tie-Loss sign test: It is a well-known pairwise statistical test to evaluate whether there is a statistical difference between the outcomes of two models. In this test, it is assumed that the two models are equivalent under the null hypothesis, i.e., each model

Table 4.9: Differences in the rankings of the non-linear proposed Wave-SVM against baseline models on UCI and KEEL datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
C-SVM [6]	3.37	1.57	Yes
Pin-SVM [27]	2.85	1.05	Yes
LINEX-SVM [31]	4.16	2.36	Yes
FP-SVM [99]	2.53	0.73	No
Wave-SVM†	1.8	-	N/A

† represents the proposed model.

Table 4.10: Differences in the rankings of the non-linear proposed Wave-TSVM against baseline models on UCI and KEEL datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
TSVM [10]	2.78	0.78	No
Pin-GTSVM [88]	3.55	1.55	Yes
IF-RVFL [101]	3.75	1.75	Yes
SLTSVM [89]	2.92	0.92	No
Wave-TSVM†	2	-	N/A

† represents the proposed model.

Table 4.11: Non-linear Wave-SVM results of pairwise Win-Tie-Loss test on UCI and KEEL datasets. Here, in [x/y/z], x, y, and z correspond to the count of wins, ties, and losses of the row model over column method, respectively.

	C-SVM [6]	Pin-SVM [27]	LINEX-SVM [31]	FP-SVM [99]
Pin-SVM [27]	[13, 28, 1]			
LINEX-SVM [31]	[5, 12, 25]	[4, 10, 28]		
FP-SVM [99]	[19, 17, 6]	[14, 18, 10]	[29, 6, 7]	
Wave-SVM†	[28, 11, 3]	[25, 12, 5]	[32, 7, 3]	[27, 12, 3]

† represents the proposed model.

Table 4.12: Non-linear Wave-TSVM results of pairwise Win-Tie-Loss test on UCI and KEEL datasets. Here, in [x/y/z], x, y, and z correspond to the count of wins, ties, and losses of the row model over column method, respectively.

	TSVM [10]	Pin-GTSVM [88]	IF-RVFL [101]	SLTSVM [89]
Pin-GTSVM [88]	[10, 4, 18]			
IF-RVFL [101]	[6, 5, 21]	[15, 2, 15]		
SLTSVM [89]	[11, 5, 16]	[21, 2, 9]	[19, 4, 9]	
Wave-TSVM†	[19, 8, 5]	[22, 3, 7]	[26, 3, 3]	[17, 10, 5]

† represents the proposed model.

wins on $D/2$ out of D datasets. Two models are considered significantly different at 5% level of significance if one model has at least $D/2 + 1.96\sqrt{D}/2$ wins. If there is a tie, the score is shared equally among the two models. For Wave-SVM, we have $D = 42$, $D/2 + 1.96\sqrt{D}/2 = 27.35$. Thus, two SVM type models are statistically different, if one of them wins on at least 27 datasets. Table 4.11 presents the pairwise Win-Tie-Loss performance of the proposed Wave-SVM and the baseline models. It has been determined that the proposed Wave-SVM has 28 (*w.r.t.* C-SVM), 25 (*w.r.t.* Pin-SVM), 32 (*w.r.t.* LINEX-SVM), and 27 (*w.r.t.* FP-SVM) wins out of 42 datasets. For Wave-TSVM, we have $D = 32$, $D/2 + 1.96\sqrt{D}/2 = 21.54$. Thus, two models are statistically different, if one of them wins on at least 21 datasets. Table 4.12 showcases the Win-Tie-Loss results of the proposed Wave-TSVM and the baseline models. It can be seen that the proposed Wave-TSVM has 19 (*w.r.t.* TSVM), 22 (*w.r.t.* Pin-GTSVM), 26 (*w.r.t.* IF-RVFL), and 17 (*w.r.t.* SLTSVM) wins out of 32 datasets. After sharing the ties equally among the models, it can be established that the proposed Wave-SVM and Wave-TSVM models perform superiorly over the baseline models.

4.5 Chapter Summary

This chapter emphasized the pivotal role of the loss function within the realm of supervised learning, highlighting how the choice of the appropriate loss function significantly influences the proficiency of the developed models. It introduced an innovative asymmetric loss function named wave loss. This novel loss function is distinguished by its aptitude for handling outliers, resilience against noise, bounded characteristics, and an essential smoothness property. These distinctive attributes collectively position the wave loss function to effectively address the complex challenges posed by real-world data. Further, by showing the possession of classification-calibrated property, the chapter laid the foundation

for its broader adoption in real-world applications. Additionally, the amalgamation of the proposed wave loss into the least squares setting of SVM and TSVM led to the development of two innovative models: Wave-SVM and Wave-TSVM, respectively. These models offer a robust and smooth alternative to existing classifiers, contributing to the advancement of classification techniques. Notably, the incorporation of the Adam algorithm for optimizing the Wave-SVM brings further innovation, as this is the first time that Adam has been utilized to solve an SVM model. This strategic use of Adam not only improves the efficiency of Wave-SVM but also offers a solution for handling large-scale problems. To tackle the optimization problem of Wave-TSVM, the chapter utilizes an iterative algorithm that requires matrix inversion at each iteration, which is intractable for large datasets. In the future, one can reformulate the Wave-TSVM to circumvent the need to compute matrix inversion. The empirical results garnered from a diverse spectrum of UCI and KEEL datasets (with and without feature noise) undeniably affirm the efficacy of the proposed Wave-SVM and Wave-TSVM against baseline algorithms.

However, we did not explore the application of the wave loss function within deep learning models, which presents an avenue for future research. Looking ahead, the crucial smoothness property inherent to the wave loss function suggests promising possibilities for its fusion with advanced machine learning and deep learning techniques. In future work, researchers can explore the fusion of the wave loss function with cutting-edge methodologies like support matrix machines to tackle complex real-world problems.

Chapter 5

HawkEye: A Robust Loss Function for Regression with Bounded, Smooth, and Insensitive Zone Characteristics

This chapter introduces the HawkEye loss function for robust support vector regression (SVR). Classical SVR relies on ε-insensitive loss, while several robust variants use bounded losses to reduce the effect of outliers and noisy observations. However, existing bounded regression losses either become non-smooth because of hard truncation or remain smooth but lack an explicit insensitive zone for ignoring small, practically negligible deviations. The HawkEye loss is designed to resolve this gap by simultaneously providing boundedness, smoothness, and an insensitive zone, thereby improving robustness and generalization in noisy regression settings. The chapter integrates HawkEye into the least-squares SVR framework to obtain the HawkEye least-squares support vector regressor (HE-LSSVR), and solves the resulting optimization problem using adaptive moment estimation (Adam) in an inverse-free manner. The proposed model is evaluated on UCI, synthetic, time-series, and brain-age datasets, demonstrating improved generalization performance and training efficiency compared with baseline regression models.

5.1 Introduction

Machine learning, a subset of artificial intelligence, has witnessed remarkable advancements in recent years, revolutionizing various fields through its ability to extract meaningful patterns from complex data [105]. Within the realm of machine learning, supervised learning stands as a cornerstone, where algorithms learn from labeled datasets to make predictions or infer relationships between input and output variables [2]. Among the diverse applications of supervised learning, regression emerges as a fundamental subfield dedicated to modeling the intricate connections between input variables and continuous output values [106].

Support vector regression (SVR) [5] is a powerful and versatile machine learning technique designed for solving regression problems. It belongs to the family of support vector machines (SVMs) [6], which are renowned for their effectiveness in classification tasks. SVMs have applications in several areas, such as breast cancer diagnosis [99, 107], Alzheimer's diagnosis [67, 108], EEG signal classification [90, 109], and many more. In a nutshell, SVMs are rooted in the framework of statistical learning theory, which equips them with the ability to generalize effectively to new and unseen data. However, SVR extends the capabilities of SVM to handle continuous output variables, making it a valuable tool for predictive modeling and function approximation. It has extensive applications across various domains, such as brain age prediction [110, 111], gas diffusion coefficient prediction [112], long-term stock index price prediction [113], financial market forecasting [114], estimation of bank profitability [115], and so forth.

Consider a regression training dataset $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$. In this chapter, N denotes the number of training samples and m denotes the feature dimension; these are local symbols corresponding to m and d, respectively, in the global notation of

Chapter 2. SVR estimates a function $f(x) = w^\intercal x + b$ by controlling residuals $r_i = y_i - f(x_i)$. In a loss-based unconstrained form, the regularized SVR objective can be written as

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\mathcal{L}(r_i), \tag{5.1}$$

where $C > 0$ controls the trade-off between model complexity and empirical risk. This form makes the central role of the regression loss explicit.

In this chapter, our primary focus revolves around enhancing the performance of SVR. In the realm of developing novel regression models, one critical aspect that garners significant attention is the design of appropriate loss functions. The regression losses relevant to this chapter, including ε-insensitive, least-squares, absolute, Huber, canal, ramp-insensitive, bounded least-squares, and BLINEX losses, are summarized in Table 2.4. Their common limitation is that each provides only part of the desired behavior: unbounded losses remain sensitive to large residuals, hard-truncated bounded losses are usually non-smooth, and smooth bounded losses generally do not include an explicit insensitive zone.

In real-world regression, outliers may deviate substantially from the dominant data pattern [116], and they can be especially difficult to detect in high-dimensional settings [7]. A robust regression loss should therefore limit the influence of extreme residuals. At the same time, an insensitive zone is desirable when small residuals are practically negligible, while smoothness is important for stable gradient-based optimization. The central gap is that existing losses rarely provide boundedness, smoothness, and an insensitive zone simultaneously.

Based on insights gleaned from previous research efforts, this chapter introduces the HawkEye loss, an innovative loss function meticulously designed to exhibit boundedness, smoothness, and the simultaneous presence of an insensitive zone. Subsequently, by incorporating the newly proposed HawkEye loss into the least squares version of SVR, we

propose a new robust and scalable model named HE-LSSVR. The optimization problem associated with HE-LSSVR model is effectively addressed through the utilization of the adaptive moment estimation (Adam) algorithm. The key contributions of this chapter can be succinctly summarized as follows:

- We introduce a novel development in the realm of supervised learning, the Hawk-Eye loss function, meticulously crafted to exhibit boundedness, smoothness, and an insensitive zone.

- We present a comparative analysis against SVR losses and broader robust regression families. To the best of our knowledge, within the surveyed SVR and LSSVR literature, HawkEye is the first loss function that is simultaneously bounded, smooth, and equipped with an exact insensitive zone.

- We amalgamate the HawkEye loss function into the least squares framework of SVR and formulate a robust and scalable model coined as HE-LSSVR.

- We address the optimization problem of HE-LSSVR by harnessing the Adam algorithm within an inverse-free optimization framework, known for its adaptive learning rate and efficacy in handling large-scale problems. To the best of our knowledge within the surveyed SVR literature, this is the first use of Adam to solve the resulting SVR optimization problem.

- We conduct numerical experiments on UCI, synthetic, and time-series datasets and compare HE-LSSVR with the baseline models in terms of predictive error and training time, while explicitly identifying comparisons for which the observed differences are small or not statistically significant.

The remainder of this chapter first states the chapter-specific gap in existing regression

losses, then presents the HawkEye loss, the HE-LSSVR formulation, and the optimization procedure, followed by the experimental evaluation and chapter summary.

5.2 Related Work

Existing regression losses that form the baseline landscape for this thesis include least-squares, absolute, Huber, ε-insensitive, ramp-insensitive, nonconvex least-squares, canal, bounded least-squares, and BLINEX losses. The chapter-specific point is the gap left by this landscape: unbounded losses remain sensitive to large residuals, hard-truncated bounded losses are usually non-smooth, and smooth bounded losses generally do not include an explicit insensitive zone. HawkEye is designed to close this gap by combining boundedness, smoothness, and an insensitive zone in a single regression loss.

5.3 HawkEye Loss and HE-LSSVR

In this section, we present the proposed HawkEye loss function, discuss its key properties, compare it with baseline loss functions, introduce the proposed HE-LSSVR model, and outline the corresponding optimization procedures.

5.3.1 HawkEye Loss Function

We present a noteworthy breakthrough in the field of supervised learning: a novel symmetric loss function termed the HawkEye loss. It is designed to manifest desirable attributes of boundedness and smoothness while concurrently incorporating an insensitive zone. Formally, the insensitive zone is characterized by a threshold parameter ε. For any prediction error $r = y - \hat{y}$, where y is the true value and $\hat{y}$ is the predicted value, the loss function assigns zero loss if $|r| \leq \varepsilon$. Outside this interval, the loss function starts to penalize the

prediction error. The inclusion of the insensitive zone allows the HawkEye loss to tolerate small errors, which can be particularly useful in reducing the impact of noise and minor discrepancies in the data.

The mathematical expression of the proposed HawkEye loss function is articulated as follows:

$$\mathcal{L}_{HE}(r, a, \lambda) = \begin{cases} \lambda \left[1 - \{-a(r+\varepsilon)+1\} e^{a(r+\varepsilon)}\right], & r \leq -\varepsilon, \\ 0, & -\varepsilon < r < \varepsilon, \\ \lambda \left[1 - \{a(r-\varepsilon)+1\} e^{-a(r-\varepsilon)}\right], & r \geq \varepsilon, \end{cases} \tag{5.2}$$

where $\varepsilon > 0$ is an insensitivity parameter that serves to establish a zone of tolerance within the loss function, $a \in \mathbb{R}^+$ is a shape parameter that determines the curvature and steepness of the loss function, and $\lambda \in \mathbb{R}^+$ is a bounding parameter that acts as a constraint to ensure that the loss does not exceed a specified bound. To visually depict the proposed HawkEye loss function (5.2), Figure 5.1 summarizes the loss and its derivative, while Figures 5.1a and 5.1b show the loss for various combinations of the parameters a and λ. The Proposed HawkEye loss function (5.2), as delineated in this work, exhibits the following intrinsic properties:

Property 1: $\mathcal{L}_{HE}(\cdot)$ exhibits an insensitive zone, meaning it assigns zero loss to residuals within a specified range, effectively ignoring minor deviations from the target value.

Property 2: $\mathcal{L}_{HE}(\cdot)$ is symmetric, signifying its impartial allocation of equal loss to deviations on both sides of the reference point.

Property 3: $\mathcal{L}_{HE}(\cdot)$ is non-negative, ensuring that the loss is always greater than or equal to zero.

Property 4: $\mathcal{L}_{HE}(\cdot)$ is zero at origin and increases monotonically with respect to $|r|$, i.e.,

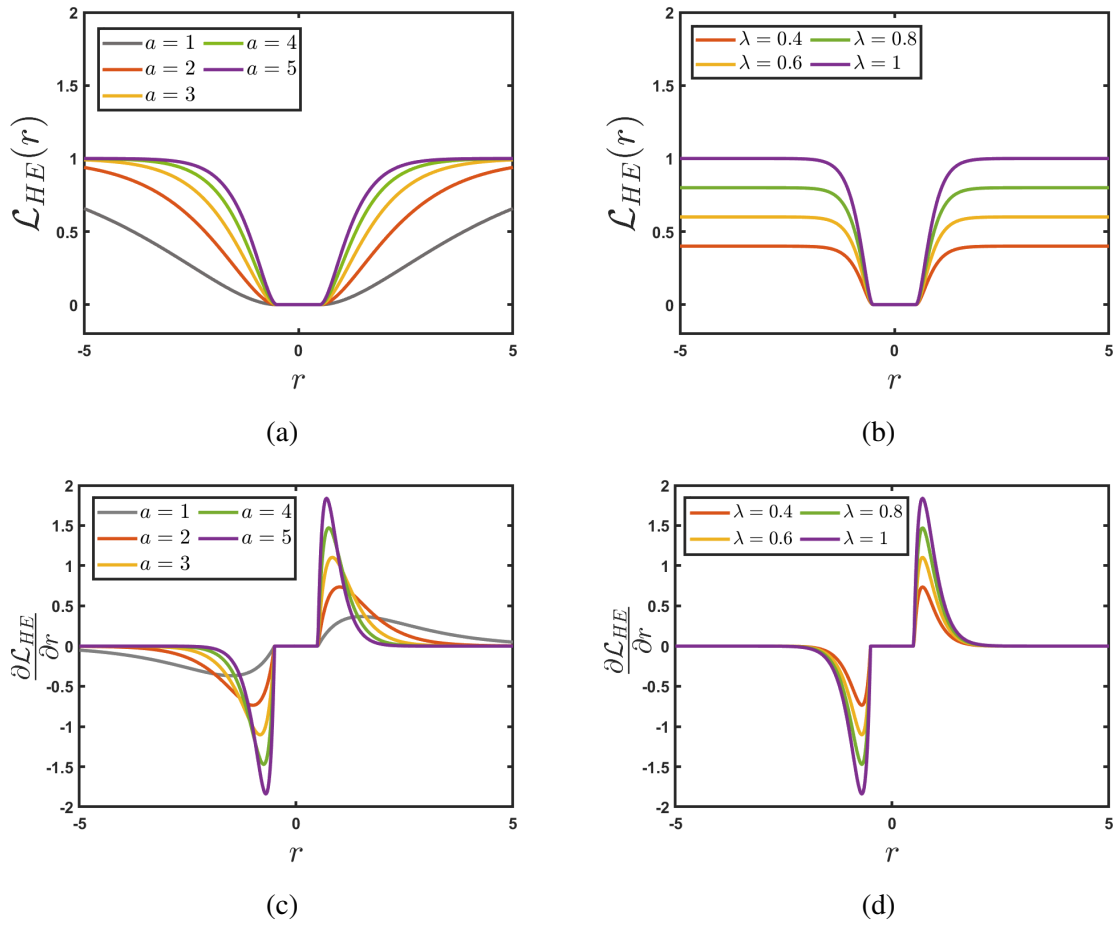


Figure 5.1: Illustration of the HawkEye loss function and its gradient for varying values of a and λ. The loss function plots demonstrate that for a fixed $\lambda = 1$ and $\varepsilon = 0.5$, increasing a results in a steeper transition beyond the insensitive zone, making the function more sensitive to larger residuals. Conversely, for a fixed $a = 5$ and $\varepsilon = 0.5$, increasing λ raises the upper bound of the loss, effectively constraining the maximum penalty for large errors. The gradient plots further illustrate these effects, where a higher a leads to sharper changes in gradient magnitude, reinforcing stronger penalization for deviations, while increasing λ scales the gradient without altering its fundamental behavior. These visualizations highlight how adjusting a and λ enables fine-tuning of the loss function, balancing robustness and sensitivity to optimize performance in different tasks.

$\mathcal{L}_{HE}(0) = 0$ and $\frac{\partial \mathcal{L}_{HE}}{\partial |r|} \geq 0$.

Property 5: $\mathcal{L}_{HE}(\cdot)$ is bounded by λ, ensuring that the loss does not grow unbounded as the error increases, thereby enhancing robustness.

Property 6: $\mathcal{L}_{HE}(\cdot)$ is smooth, indicating that it is differentiable and suitable for gradient-based optimization algorithms.

5.3.2 Proofs of HawkEye Loss Properties

5.3.3 Proof of the Properties of the HawkEye Loss Function

In this subsection, we provide the detailed proof of the properties of the HawkEye loss function.

Property 1: $\mathcal{L}_{HE}(\cdot)$ exhibits an insensitive zone, meaning it assigns zero loss to residuals within a specified range, effectively ignoring minor deviations from the target value.

Proof. For residuals r, where $-\varepsilon < r < \varepsilon$, the loss function is defined as $\mathcal{L}_{HE}(r) = 0$. This is clearly stated in the definition of the loss function.

This means that any data points with residuals within this range (i.e., $r \in (-\varepsilon, \varepsilon)$) do not contribute to the overall loss. As a result, the loss remains unaffected by small deviations within this zone. Since the gradient of the loss function with respect to these points is also zero, these data points do not influence the optimization process. In other words, the ε-insensitive zone essentially ignores data points whose residuals fall within this range, reducing the number of training points that contribute to the final model. This results in a sparse representation, as only the points outside the ε-insensitive zone (where $r \geq \varepsilon$ or $r \leq -\varepsilon$) actively affect the model. □

Property 2: $\mathcal{L}_{HE}(\cdot)$ is symmetric, signifying its impartial allocation of equal loss to deviations on both sides of the reference point.

Proof. Consider two values r and $-r$. From the definition, we can easily show that: $\mathcal{L}_{HE}(r) = \mathcal{L}_{HE}(-r)$. This symmetry property is a direct consequence of the construction of the loss function. □

Property 3: $\mathcal{L}_{HE}(\cdot)$ is non-negative, ensuring that the loss is always greater than or equal to zero.

Proof. For all values of r, it can be seen easily that all the terms in the definition of $\mathcal{L}_{HE}(\cdot)$ are non-negative. Hence, ensuring the non-negativity property. □

Property 4: $\mathcal{L}_{HE}(\cdot)$ is zero at origin and increases monotonically with respect to $|r|$, i.e., $\mathcal{L}_{HE}(0) = 0$ and $\frac{\partial \mathcal{L}_{HE}}{\partial |r|} \geq 0$.

Proof. Since $\varepsilon > 0$, the origin will always lie in the region $-\varepsilon < r < \varepsilon$. Then, from the definition, we can easily see that $\mathcal{L}_{HE}(0) = 0$. Now, to prove that the HawkEye loss function increases monotonically with respect to $|r|$, let us analyze each region separately.

For $r \leq -\varepsilon$:

We can write $\mathcal{L}_{HE}(|r|) = \lambda \left[1 - \{-a(-|r| + \varepsilon) + 1\} e^{a(-|r|+\varepsilon)}\right]$.

Taking the derivative with respect to $|r|$, we get:

$\frac{\partial \mathcal{L}_{HE}}{\partial |r|} = -\lambda a^2(-|r| + \varepsilon) e^{a(-|r|+\varepsilon)}$. Since λ, $a > 0$, $-|r| + \varepsilon \leq 0$, and $e^{a(-|r|+\varepsilon)}$ is always positive, then we can surely say that the derivative is always non-negative.

For $-\varepsilon < r < \varepsilon$:

In this region, the HawkEye loss function is constant, so $\frac{\partial \mathcal{L}_{HE}}{\partial |r|} = 0$.

For $r \geq \varepsilon$:

We can write $\mathcal{L}_{HE}(|r|) = \lambda \left[1 - \{a(|r| - \varepsilon) + 1\} e^{-a(|r|-\varepsilon)}\right]$.

Taking the derivative with respect to $|r|$, we get:

$\frac{\partial \mathcal{L}_{HE}}{\partial |r|} = \lambda a^2(|r| - \varepsilon) e^{-a(|r|-\varepsilon)}$. Since λ, $a > 0$, $|r| - \varepsilon \geq 0$, and $e^{-a(|r|-\varepsilon)}$ is always positive, then the derivative is always non-negative.

In conclusion, the HawkEye loss function increases monotonically with respect to $|r|$, i.e., $\frac{\partial \mathcal{L}_{HE}}{\partial |r|} \geq 0$.

□

Property 5: $\mathcal{L}_{HE}(\cdot)$ is bounded by λ, ensuring that the loss does not grow unbounded as the error increases, thereby enhancing robustness.

Proof. For this, we analyze its behavior in the three defined regions.

For $r \leq -\varepsilon$:

In this region, we have:

$$\mathcal{L}_{HE}(r) = \lambda \left[1 - \{-a(r+\varepsilon) + 1\} e^{a(r+\varepsilon)}\right]. \tag{5.3}$$

To prove that $\mathcal{L}_{HE}(r)$ is bounded by λ, we need to show that the expression in the square bracket of equation (5.3) is less than or equal to 1, i.e., $\{-a(r+\varepsilon) + 1\} e^{a(r+\varepsilon)} \geq 0$.

Let's simplify the expression.

$e^{a(r+\varepsilon)}$ is a positive term.

$-a(r+\varepsilon) + 1 \geq 0$ (since $a > 0$ and $r \leq -\varepsilon$).

Thus, the expression in the square bracket of equation (5.3) is less than or equal to 1. Therefore, in this region, $\mathcal{L}_{HE}(r)$ is bounded by λ.

For $-\varepsilon < r < \varepsilon$:

In this region, we have $\mathcal{L}_{HE}(r) = 0$, which is trivially bounded by λ.

For $r \geq \varepsilon$:

Similar to the first region, we can easily demonstrate that in this region $\mathcal{L}_{HE}(r)$ is also bounded by λ.

Hence, in all three regions, we have shown that $\mathcal{L}_{HE}(r)$ is bounded by λ for the specified parameter values $a > 0$, $\lambda > 0$, and $\varepsilon > 0$. □

Property 6: $\mathcal{L}_{HE}(\cdot)$ is smooth, indicating that it is differentiable and suitable for gradient-based optimization algorithms.

Proof. The loss function $\mathcal{L}_{HE}(\cdot)$ is meticulously constructed by combining continuous functions and exponential terms, ensuring its overall differentiability. Nevertheless, to dispel any concerns regarding its differentiability specifically at $r = \varepsilon$ and $r = -\varepsilon$, we provide a thorough demonstration limited to these particular points.

At $r = \varepsilon$:

$$\begin{aligned}
\text{L.H.D.} &= \lim_{h\to 0} \frac{\mathcal{L}_{HE}(\varepsilon - h) - \mathcal{L}_{HE}(\varepsilon)}{-h} \\
&= \lim_{h\to 0} \frac{0 - 0}{-h} && \text{(Using the definition)} \\
&= 0.
\end{aligned}$$

$$\begin{aligned}
\text{R.H.D.} &= \lim_{h\to 0} \frac{\mathcal{L}_{HE}(\varepsilon + h) - \mathcal{L}_{HE}(\varepsilon)}{h} \\
&= \lim_{h\to 0} \frac{\lambda\left[1 - \{a(\varepsilon + h - \varepsilon) + 1\}e^{-a(\varepsilon+h-\varepsilon)}\right] - 0}{h} && \text{(Using the definition)} \\
&= \lim_{h\to 0} \frac{\lambda\left[1 - \{ah + 1\}e^{-ah}\right]}{h} && (\tfrac{0}{0} \text{ form}) \\
&= \lim_{h\to 0} -\lambda\left[-a(ah + 1)e^{-ah} + ae^{-ah}\right] && \text{(Using L'hospital rule)} \\
&= -\lambda\left[-a + a\right] \\
&= 0.
\end{aligned}$$

Thus, at $r = \varepsilon$, the left-hand derivative and right-hand derivative are both equal. Hence, $\mathcal{L}_{HE}(\cdot)$ is differentiable at $r = \varepsilon$.

At $r = -\varepsilon$:

$$\begin{aligned}
\text{L.H.D.} &= \lim_{h\to 0} \frac{\mathcal{L}_{HE}(-\varepsilon - h) - \mathcal{L}_{HE}(-\varepsilon)}{-h} \\
&= \lim_{h\to 0} \frac{\lambda\left[1 - \{-a(-\varepsilon - h + \varepsilon) + 1\}e^{a(-\varepsilon - h + \varepsilon)}\right] - 0}{-h} && \text{(Using the definition)} \\
&= \lim_{h\to 0} \frac{\lambda\left[1 - \{-a(-h) + 1\}e^{a(-h)}\right]}{-h} && (\tfrac{0}{0} \text{ form}) \\
&= \lim_{h\to 0} \lambda\left[-a(ah+1)e^{-ah} + ae^{-ah}\right] && \text{(Using L'hospital rule)} \\
&= \lambda\left[-a + a\right] \\
&= 0.
\end{aligned}$$

$$\begin{aligned}
\text{R.H.D.} &= \lim_{h\to 0} \frac{\mathcal{L}_{HE}(-\varepsilon + h) - \mathcal{L}_{HE}(-\varepsilon)}{h} \\
&= \lim_{h\to 0} \frac{0 - 0}{h} && \text{(Using the definition)} \\
&= 0.
\end{aligned}$$

Thus, at $r = -\varepsilon$, the left-hand derivative and right-hand derivative are both equal. Hence, $\mathcal{L}_{HE}(\cdot)$ is differentiable at $r = -\varepsilon$. Consequently, we have established that $\mathcal{L}_{HE}(\cdot)$ is differentiable throughout its domain. □

To enable gradient-based optimization, we can derive the derivative of $\mathcal{L}_{HE}(\cdot)$ with respect to r:

$$\frac{\partial \mathcal{L}_{HE}(r, a, \lambda)}{\partial r} = \begin{cases} \lambda a^2 (r + \varepsilon) e^{a(r+\varepsilon)}, & r < -\varepsilon, \\ 0, & -\varepsilon \le r \le \varepsilon, \\ \lambda a^2 (r - \varepsilon) e^{-a(r-\varepsilon)}, & r > \varepsilon. \end{cases} \tag{5.4}$$

The derivative of $\mathcal{L}_{HE}(\cdot)$ is visualized for different values of parameters in Figures 5.1c and 5.1d. The shape of the derivative provides valuable insights into our loss function and

its behavior during minimization using gradient descent or some related method. For all values of a and λ, the derivative is 0 when $|r| \leq \varepsilon$, indicating no sensitivity to minor residuals. However, as the magnitude of the residual exceeds ε, the derivative increases with the growth of a and λ. Further, as the magnitude of the residual progressively increases beyond a certain threshold, the magnitude of the derivative begins to decrease, ultimately converging toward zero. Consequently, as the magnitude of an outlier's residual grows, its influence on the gradient descent process diminishes, and once it surpasses a certain threshold, the outlier's impact becomes nearly negligible.

Table 5.1 summarizes the main structural properties of the regression losses used for comparison with HawkEye.

Table 5.1: Demonstrate the characteristics of baseline loss functions used in regression and the proposed HawkEye loss.

Loss function ↓\Characteristics →	**Robust**	**Insensitive zone**	**Bounded**	**Convex**	**Smooth**
Least square loss	✗	✗	✗	✓	✓
Absolute loss	✗	✗	✗	✓	✗
Huber loss	✗	✗	✗	✓	✓
Insensitive loss	✗	✓	✗	✓	✗
Ramp insensitive loss	✓	✓	✓	✗	✗
Non-convex least square loss	✓	✗	✓	✗	✗
Ramp insensitive least square loss	✓	✓	✓	✗	✗
Quadratic non-convex insensitive loss	✓	✓	✓	✗	✗
Canal loss	✓	✓	✓	✗	✗
Bounded least square loss	✓	✗	✓	✗	✓
BLINEX loss	✓	✗	✓	✗	✓
HawkEye loss (Proposed)	✓	✓	✓	✗	✓

We next present a comparative analysis of the proposed HawkEye loss function alongside several established baseline loss functions.

5.3.4 Comparative Analysis of HawkEye Loss Function with Baseline Loss Functions

In this subsection, we conduct a comparative analysis of the proposed HawkEye loss function with some established baseline loss functions. Each baseline loss function is examined individually to highlight its characteristics and performance relative to the HawkEye

loss function. For the sake of clarity, we presented visual depictions in Figure 5.2.

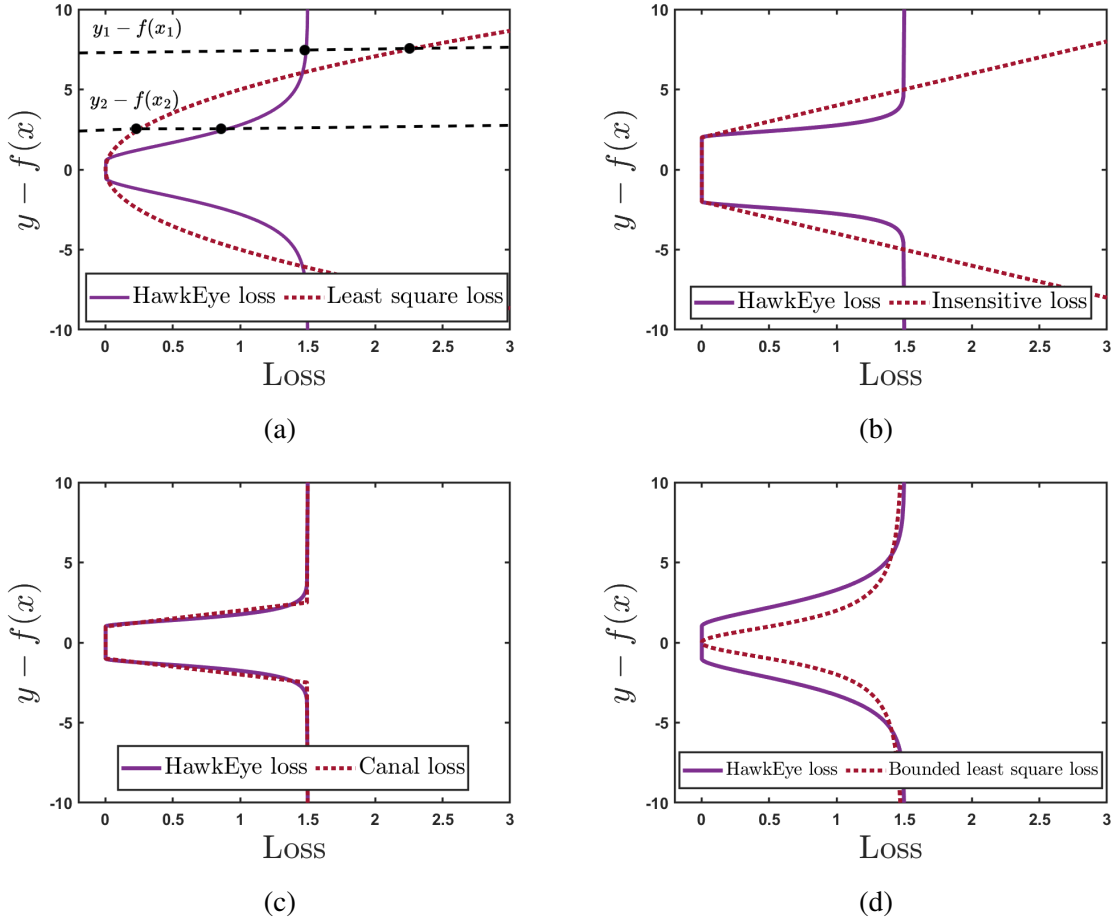


Figure 5.2: Visual comparison of HawkEye loss and baseline loss functions. Subfigures (a), (b), (c), and (d) demonstrate the comparison of HawkEye loss function with the least square loss, insensitive loss, canal loss, and bounded least square loss, respectively.

1. **HawkEye loss** ($\mathcal{L}_{HE}$) *vs* **Least square loss** ($\mathcal{L}_{ls}$): Consider a data point (x_1, y_1) with a noisy label where the target value y_1 is mislabeled for some reason. As a consequence, the target value y_1 deviates significantly from the actual model prediction $f(x_1)$, resulting in a large absolute difference of $|y_1 - f(x_1)|$. In such cases, the least squares loss becomes exceedingly large, and minimizing this loss would have a notorious repercussion on the overall performance of the model. In contrast, when we apply the HawkEye loss to such a data point in similar circumstances, it tends to

approach a constrained value, and we note that $\mathcal{L}_{HE}(x_1)$ is notably less than $\mathcal{L}_{ls}(x_1)$ (see Figure 5.2a). This observation underscores that the HawkEye loss function effectively mitigates the influence of noisy data points, highlighting its robustness in the presence of such outliers. Now, let's turn our attention to another data point (x_2, y_2) that is clean and exhibits a small absolute difference $|y_2 - f(x_2)|$. Such points are inherently more amenable to misestimation and thus warrant closer attention. Notably, when we examine the least squares loss for this point, it is less than the corresponding HawkEye loss, as evidenced by $\mathcal{L}_{HE}(x_2) > \mathcal{L}_{ls}(x_2)$. Thus, it can be confidently asserted that, with appropriately selected parameters, the HawkEye loss function not only demonstrates resilience in the presence of noisy data points but also exhibits a capacity for affording greater attention to clean, non-outlier data points.

2. **HawkEye loss** ($\mathcal{L}_{HE}$) *vs* **Insensitive loss** ($\mathcal{L}_{\varepsilon}$): Similar to the least squares loss function, the insensitive loss function imposes a substantial penalty on samples exhibiting large deviations, making it less robust in the presence of outliers when compared to the HawkEye loss (see Figure 5.2b). Furthermore, for a judiciously chosen parameter value, the HawkEye loss exhibits a propensity to allocate greater emphasis to non-outlier points compared to the insensitive loss function. It is also important to highlight that the HawkEye loss function surpasses the insensitive loss function in terms of smoothness. The inherent smoothness property of the HawkEye loss is an added advantage in optimization processes.

3. **HawkEye loss** ($\mathcal{L}_{HE}$) *vs* **Canal loss** ($\mathcal{L}_{canal}$): The Canal loss is characterized by its capability to constrain the loss to a predefined value through the selection of parameters θ and ε. Similarly, the HawkEye loss function shares this property, as it determines an upper bound λ for penalty and restricts the loss to stop raising after a

certain deviation, as illustrated in Figure 5.2c. However, in the context of non-outlier data points, the canal loss maintains a fixed shape. In contrast, the proposed HawkEye loss function exhibits greater versatility, as it can adapt its shape to suit diverse situations by adjusting the shape parameter value a. It is also essential to underscore that the HawkEye loss outshines the canal loss not only in adaptability but also in terms of smoothness. Hence the HawkEye loss, with its smooth profile, presents a more advantageous characteristic compared to the canal loss.

4. **HawkEye loss** ($\mathcal{L}_{HE}$) *vs* **Bounded least square loss** ($\mathcal{L}_{bls}$): The bounded least square loss function is equipped with the capability to confine the loss within a predefined threshold by adjusting the parameter t. Simultaneously, it offers the flexibility to adapt its shape for non-outlier data points through the modulation of the parameter θ (see Figure 5.2d). Intriguingly, the HawkEye loss function also shares both of these remarkable characteristics. A notable highlight is that the HawkEye loss introduces an additional dimension by incorporating the concept of an insensitive zone. Within the SVR and LSSVR losses surveyed in Chapter 2.4, HawkEye is distinguished by jointly providing smoothness, boundedness, and an exact insensitive zone.

5. **HawkEye loss** ($\mathcal{L}_{HE}$) *vs* **BLINEX loss** ($\mathcal{L}_{BLINEX}$): BLINEX is an asymmetric loss function that employs a soft truncation strategy, making it both smooth and bounded. This characteristic allows it to handle outliers effectively while maintaining computational efficiency. However, BLINEX lacks an insensitive zone, which limits its ability to disregard minor deviations and noise, potentially leading to overfitting in noisy datasets. In contrast, the HawkEye loss function is symmetric, treating errors on both sides equally, and integrates three essential properties: smoothness, boundedness, and an insensitive zone. Like BLINEX, HawkEye retains the benefits of smoothness and

boundedness but extends its versatility by incorporating an insensitive zone, making it a more comprehensive solution for improving model robustness and generalization in the presence of noisy data.

Traditional loss functions, such as least square loss, absolute loss, Huber loss, and insensitive loss, are unbounded and lack robustness. The broader review in Section 2.4.4 also shows that redescending M-estimators can be bounded and smooth but do not contain an exact interval of zero loss. To the best of our knowledge, within the surveyed SVR and LSSVR literature, the proposed HawkEye loss is the first loss that is bounded, smooth, and simultaneously possesses an exact insensitive zone.

5.3.5 SVR with HawkEye Loss Function

In this subsection, we incorporate the newly proposed HawkEye loss function into the least squares framework of SVR and establish a novel support vector regressor that manifests robustness against noise and outliers and desirable smoothness characteristics. We refer to this advanced regressor as the HE-LSSVR. For the sake of simplicity and without loss of generality, we use the terminology w for $[w^\intercal, b]$ and x_i for $[x_i, 1]^\intercal$ throughout this chapter. The formulation of the proposed HE-LSSVR is given as:

$$\begin{aligned} \underset{w,\xi}{min} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\mathcal{L}_{HE}(\xi_i), \\ \text{subject to} \quad & \xi_i = y_i - \left(w^\intercal\phi(x_i)\right),\ \forall\, i = 1, 2, \ldots, N, \end{aligned} \tag{5.5}$$

where $\frac{1}{2}\|w\|^2$ is the regularizer term, $\sum_{i=1}^{N}\mathcal{L}_{HE}(\xi_i)$ is the empirical risk corresponds to the proposed HawkEye loss function, $C > 0$ is a regularization parameter, ξ_i indicates the regression error of ith input, and $\phi(\cdot)$ represents the high-dimensional feature mapping associated with the kernel function. Let us assume that $\mathcal{K} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a Mercer

kernel and $H_{\mathcal{K}}$ is a reproducing kernel Hilbert space induced by $\mathcal{K}$. In practice, kernel functions are often employed to address nonlinear problems by formulating the relevant dual problems. However, due to the non-convex nature of equation (5.5), this approach can be quite formidable in our scenario. In this case, to enhance the capacity of HE-LSSVR for nonlinear adaptation, we use the representer theorem [70], which allows us to express w in equation (5.5) as follows:

$$w = \sum_{k=1}^{N} \alpha_k \phi(x_k), \tag{5.6}$$

where $\alpha = (\alpha_1, \ldots, \alpha_N)^\intercal$ is the coefficient vector. We note that the minimization problem (5.5) works in a reproducing kernel Hilbert space, and the penalty term is non-decreasing, so one can apply the representer theorem to the optimization problem (5.5).
Substituting the value of w from (5.6) into (5.5), the primal optimization problem (5.5) can be transformed into the following unconstrained optimization problem:

$$\underset{\alpha}{min} \sum_{i=1}^{N} \sum_{k=1}^{N} \frac{1}{2} \alpha_i \alpha_k \mathcal{K}(x_i, x_k) + C \sum_{i=1}^{N} \mathcal{L}_{HE}(\xi_i), \tag{5.7}$$

where $\xi_i = y_i - \sum_{k=1}^{N} \alpha_k \mathcal{K}(x_i, x_k)$ and $\mathcal{K}(x_i, x_k) = \phi(x_i) \cdot \phi(x_k)$ is the kernel function.

5.3.6 Adaptive Moment Estimation (Adam) Algorithm for HE-LSSVR

The non-convex nature of the HawkEye loss function poses challenges in optimization, as the optimization problem for the proposed HE-LSSVR becomes non-convex. This complexity prohibits the use of convex optimization techniques to solve the model (5.7). It is also noteworthy to acknowledge that non-convex and non-smooth optimization problems are known to be computationally challenging, often requiring specialized techniques for

optimization such as the branch and bound algorithm [33], the convex-concave procedure (CCCP) [117], and so on, which are not suitable for large-scale problems. To overcome the difficulties posed by the non-convexity of the loss function, we have designed the HawkEye loss function to be smooth. By ensuring the smoothness of the loss function, we are able to utilize gradient-based optimization algorithms to optimize the model. Leveraging gradient-based optimization offers several advantages, notably expediting convergence during the training process. This is particularly valuable since gradient-based algorithms typically exhibit faster convergence compared to quadratic programming solvers [59]. In this chapter, we employ the adaptive moment estimation (Adam) [23] optimization technique to address the HE-LSSVR problem (5.7). It effectively combines the strengths of two popular optimization algorithms, AdaGrad [60] and RMSProp [100]. There are mainly three reasons to adopt Adam algorithm. Firstly, Adam employs adaptive learning rates for each weight parameter. This adaptability ensures that the learning rate is neither too high, which could lead to convergence issues, nor too low, which could slow down convergence. Thus contributing to faster and more stable convergence. Secondly, Adam incorporates the concept of momentum, which improves convergence in complex optimization landscapes. By maintaining a moving average of past gradients, it navigates these landscapes more efficiently and accelerates the convergence process. Finally, Adam utilizes exponentially decreasing averages for previous gradients and squared gradients, providing flexibility to adapt to changing gradient behavior. This feature is well-suited for non-convex optimization tasks, where gradient characteristics may vary significantly throughout the training process.

Now, to begin with, let the objective function of (5.7) be $H(\alpha)$. Then, the procedure for applying the Adam algorithm is as follows:

1. **Initialization:** Start by initializing the algorithm with appropriate values for hyperparameters:

Learning Rate (γ): This is the local step size for updating model parameters in the Adam procedure; it is unrelated to the signed-margin notation fixed in Chapter 2. β_1 and β_2: These are constants for computing moving averages of gradients and squared gradients. δ: A small constant added to prevent division by zero.

2. **Initialize Moments:** Initialize the values of the first moment estimate (m) and the second moment estimate (v). These moments are used to keep track of past gradients and squared gradients, respectively.

3. **Iterative Optimization:** At each iteration t:

   - Compute the gradient of the objective function $H(\alpha)$ with respect to the model parameters α.

$$\nabla H(\alpha_t) = \mathcal{K}\alpha + C\sum_{i=1}^{N} \frac{\partial \mathcal{L}_{HE}}{\partial \alpha}, \tag{5.8}$$

   where

$$\frac{\partial \mathcal{L}_{HE}}{\partial \alpha} = \begin{cases} -\lambda a^2(\xi_i + \varepsilon)e^{a(\xi_i+\varepsilon)}\mathcal{K}_i, & \xi_i < -\varepsilon, \\ 0, & -\varepsilon \leq \xi_i \leq \varepsilon, \\ -\lambda a^2(\xi_i - \varepsilon)e^{-a(\xi_i-\varepsilon)}\mathcal{K}_i, & \xi_i > \varepsilon. \end{cases}$$

   Here $\mathcal{K}_i$ represent the ith row of the kernel matrix $\mathcal{K}$.

   - Update the first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\,\nabla H(\alpha_t). \tag{5.9}$$

- Update the second moment estimate:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla H(\alpha_t)^2. \tag{5.10}$$

Algorithm 5.1 Adam algorithm for HE-LSSVR

Input:
The dataset: $\{x_i, y_i\}_{i=1}^N$, $y_i \in \mathbb{R}$;
The parameters: Regularization parameter C, HawkEye loss parameters λ, a, and ε, mini-batch size s, decay rates β_1 and β_2, learning rate γ, constant δ, maximum iteration number T;
Initialize: α_0 and t;
Output:
The classifiers parameters: α;
1 : Select s samples $\{x_i, y_i\}_{i=1}^s$ uniformly at random.
2 : Computing ξ_i :

$$\xi_i = y_i - \sum_{k=1}^{s} \alpha_k \mathcal{K}(x_i, x_k), \quad i = 1, \ldots, s; \tag{5.11}$$

3 : Compute $\nabla H(\alpha_t)$: (5.8);
4 : Compute m_t: (5.9);
5 : Compute v_t: (5.10);
6 : Compute $\hat{m}_t$: (5.12);
7 : Compute $\hat{v}_t$: (5.13);
8 : Updating solution α_t: (5.14);
9 : Updating current iteration number: $t = t + 1$.
Until:
$t = T$
Return: α_t.

- Correct the bias in moments (due to initialization):

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)}, \tag{5.12}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)}. \tag{5.13}$$

- Update model parameter:

$$\alpha_t = \alpha_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \delta}}. \tag{5.14}$$

4. **Convergence Criterion:** Monitor the convergence of the optimization process. A common stopping criterion is a maximum number of iterations or a target value for the objective function $H(\alpha)$.

5. **Termination:** When the algorithm meets the convergence criterion, terminate the optimization process, and the final model parameters α represent the optimal solution for the problem $H(\alpha)$.

The structure of the Adam algorithm to solve the HE-LSSVR model is clearly described in Algorithm 5.1. Once the optimal α is obtained, the following decision function can be utilized to predict the target value of a new sample x.

$$f(x) = \sum_{k=1}^{N} \alpha_k \, \mathcal{K}\left(x, x_k\right). \tag{5.15}$$

For delineating the intricacies of the proposed HE-LSSVR model, in Figure 5.3, we present a meticulously crafted flowchart that captures the essence of our methodology. This visual narrative serves not only as a roadmap for our methodology but also as a testament to the strategic interplay of components that positions HE-LSSVR as a cutting-edge regression model.

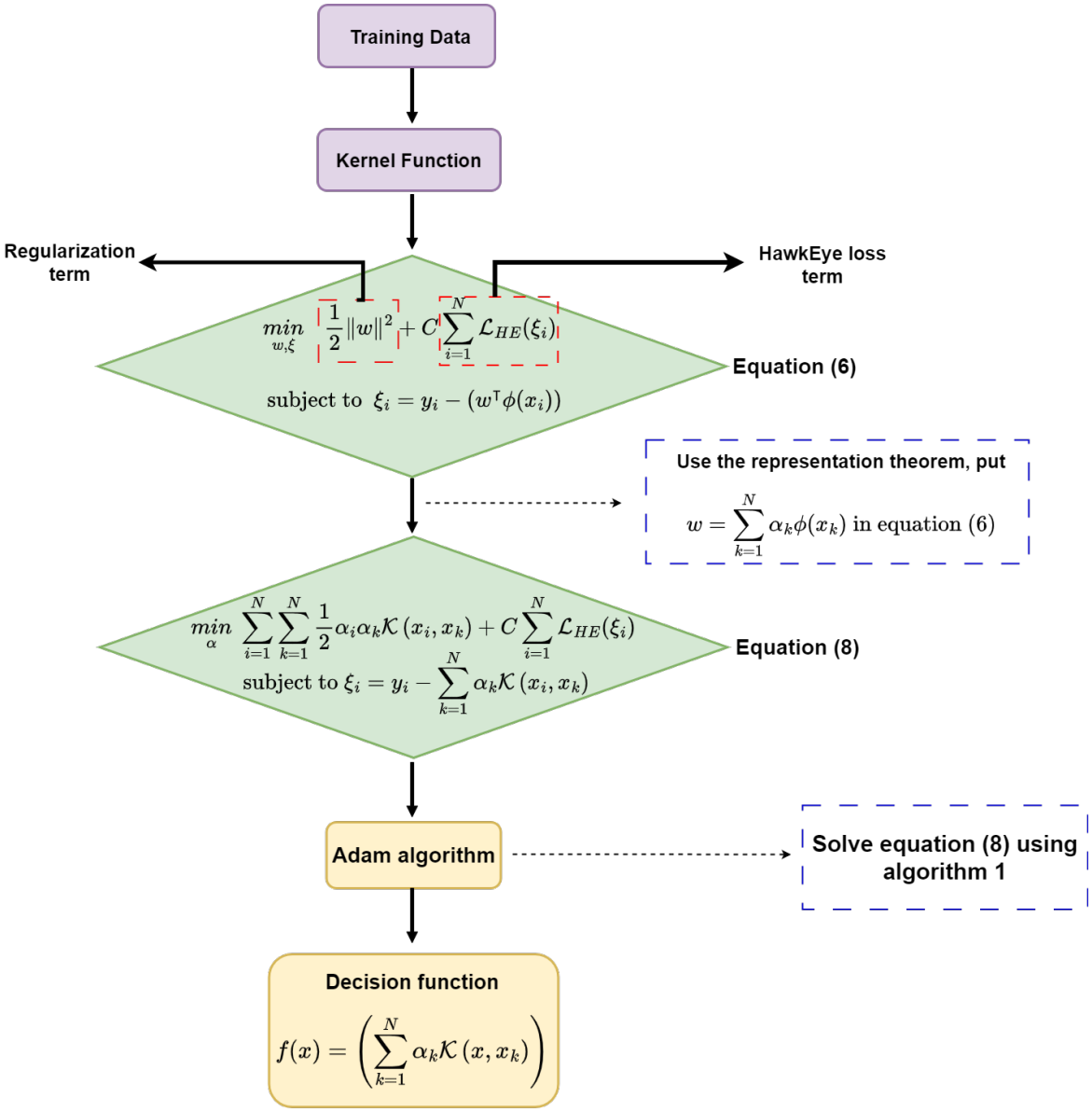


Figure 5.3: Flowchart illustrating the key stages of the HE-LSSVR model, showcasing the transformation from original input data to the decision function, highlighting the strategic integration of the HawkEye loss function and the application of representer theorem and Adam algorithm.

5.4 Numerical Experiments and Result Analysis

In this section, we conduct extensive experiments on UCI, synthetic, and time series datasets to compare the proposed HE-LSSVR model with existing state-of-the-art models, which include support vector regression (SVR) [5], least squares support vector regression (LS-SVR) [8], bounded least squares support vector regression (BLSSVR) [47], random vector functional link (RVFL) neural network [16, 118], and BLINEX support vector regression (BXSVR) [49]. The experimental setup and evaluation metrics are described below. Hyperparameter selection and performance reporting follow the same partitioning and search procedure for every baseline and proposed model, so the comparisons are controlled under a common setup. The experiments do not use a nested cross-validation loop or a validation split independent of the partitions used for reporting. Consequently, selection over the reported partitions can introduce optimistic bias [78], particularly when performance differences are small. The results should therefore be interpreted as comparative evidence under the stated protocol and not as an independent estimate of the error expected on a new external cohort.

5.4.1 Evaluation Metrics

Let y_i, $f(x_i)$ represent the actual and predicted values of x_i, respectively. To assess the models' performance, we employ four evaluation metrics: root mean squared error (RMSE), mean absolute error (MAE), positive error (Error$_{pos}$), and negative error (Error$_{neg}$). The specific definitions of these metrics are listed in Table 5.2. We primarily use RMSE as the key comparative measure, while the other metrics serve as evaluation criteria. Further, to show the efficiency of the proposed HE-LSSVR, we also performed a comparative analysis of the training time. The reported times only include the time taken for training the models with the optimized hyperparameters.

Table 5.2: The evaluation metrics.

Metrics	Definitions
RMSE	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - f\left(\mathbf{x}_i\right)\right)^2}$
MAE	$\frac{1}{N}\sum_{i=1}^{N}\left\|\left(y_i - f\left(\mathbf{x}_i\right)\right)\right\|$
Error$_{pos}$	$\frac{1}{N}\sum_{i=1, y_i \geq f(\mathbf{x}_i)}^{N}\left\|y_i - f\left(\mathbf{x}_i\right)\right\|$
Error$_{neg}$	$\frac{1}{N}\sum_{i=1, y_i < f(\mathbf{x}_i)}^{N}\left\|y_i - f\left(\mathbf{x}_i\right)\right\|$

5.4.2 Experiments on Benchmark UCI Datasets

In this subsection, we use 18 regression benchmark datasets from the UCI repository [80] to assess and compare the performance of the proposed HE-LSSVR against the baseline models. These datasets can be categorized into three groups: 1) 8 small-size datasets with samples ranging from 60 up to 500; 2) 4 medium-size datasets with samples ranging from 501 up to 1000; and 3) 6 large-size datasets with samples ranging from 1001 up to 40768. The detailed description of the datasets is provided in Table 5.3.

The experimental result of the proposed HE-LSSVR and baseline models (SVR, LS-SVR, BLSSVR, RVFL, and BXSVR) on UCI datasets in terms of metric RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ is presented in Table 5.5. The evaluation is primarily based on the RMSE, a critical metric where lower values indicate superior model performance. The HE-LSSVR model exhibits outstanding performance, achieving the lowest RMSE in 8 out of 18 datasets. Among the remaining 10 datasets, it achieves second position in 7, third position in 2, and fourth position in 1, demonstrating its consistency and robustness across diverse data. These results support the competitiveness of HE-LSSVR across the evaluated datasets, while the relatively small average gaps to BLSSVR and BXSVR require cautious interpretation under the non-nested selection protocol. The average RMSE values further validate the

Table 5.3: Description of benchmark UCI datasets.

Datasets	No. of samples	No. of features
Airfoil_Self_Noise	1503	5
Delta_Ailerons	7129	5
Forest_Fires	517	12
Yacht_Hydrodynamics	308	6
2D_Planes	40768	10
Ailerons	13750	40
Parkinsons_Telemonitoring	5875	21
Pole_Telecomm	15000	48
2014_2015 CSM dataset	185	11
cpu_pref	209	9
hungary chickenpox	522	19
istanbul stock exchange data	536	8
mcs_ds_edited_iter_shuffled	107	5
qsar_aquatic toxicity	546	8
Daily_Demand_Forecasting_Orders	60	12
bodyfat	252	14
machine	209	9
slump_test	103	10

Table 5.4: Rank of RMSE for the proposed HE-LSSVR and baseline models on benchmark UCI datasets.

Dataset	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
2014_2015 CSM dataset	4	5	3	6	1	2
cpu_pref	1	6	2.5	5	4	2.5
hungary chickenpox	4	6	2.5	5	1	2.5
istanbul stock exchange data	4	5	3	6	1	2
mcs_ds_edited_iter_shuffled	4	6	2	5	3	1
qsar_aquatic toxicity	6	5	3	4	1	2
Daily_Demand_Forecasting_Orders	3	6	4	5	1.5	1.5
bodyfat	2	6	3	5	4	1
machine	1	6	2.5	5	4	2.5
slump_test	2	5	3	6	4	1
Airfoil_Self_Noise	4	5	3	6	2	1
Delta_Ailerons	5	6	4	1	3	2
Forest_Fires	4	6	2	5	2	2
Yacht_Hydrodynamics	4	5	2	6	1	3
2D_Planes	-	5	3	1	4	2
Ailerons	5	6	3	1	2	4
Parkinsons_Telemonitoring	2	6	5	1	4	3
Pole_Telecomm	5	6	4	2	3	1
Average Rank	3.53	5.61	3.03	4.17	2.53	**2**

† represents the proposed model.

effectiveness of HE-LSSVR. The existing SVR, LS-SVR, BLSSVR, RVFL, and BXSVR exhibit average RMSE values of 0.3113, 0.6629, 0.3266, 0.3901, and 0.30, respectively. In contrast, the proposed HE-LSSVR outperforms the baseline models with an average RMSE of 0.2736. To ensure an accurate assessment of the models' performance, it is important to rank them individually for each dataset instead of relying solely on average RMSE. As exceptional performance on certain datasets may overshadow subpar performance on others. The model with the lowest RMSE receives the lowest rank and vice-versa. The average rank for SVR, LSSVR, BLSSVR, RVFL, BXSVR, and the proposed HE-LSSVR are 3.53, 5.61, 3.03, 4.17, 2.53, and 2, respectively. The lowest average rank indicates the strongest aggregate ranking for HE-LSSVR under the reported protocol; it does not by itself establish statistical superiority over every baseline. The rank corresponding to each dataset is provided in Table 5.4.

To further support the effectiveness of the proposed HE-LSSVR model, we conducted a statistical analysis using the Friedman test, followed by the Nemenyi post hoc test. The Friedman test [24] is utilized to statistically assess the significance of various models. In this test, each individual model is ranked independently for each dataset. The null hypothesis posits that all models are essentially identical, implying that the average rank of each model is equivalent. The Friedman statistic adheres to the chi-squared χ_F^2 distribution with degrees of freedom (d.f.) equal to $p-1$, where p represents the number of models. This statistic is computed as follows:

$$\chi_F^2 = \frac{12D}{p(p+1)} \left[\sum_e R_e^2 - \frac{p(p+1)^2}{4} \right]. \tag{5.16}$$

Here, D signifies the total number of datasets, and R_e denotes the mean rank of the eth model out of the p models. However, the Friedman statistic is overly cautious in nature. To

Table 5.5: RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the benchmark UCI datasets for the proposed HE-LSSVR model and the baseline models.

Dataset	Metric	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
2014_2015 CSM dataset	RMSE	0.2604	0.3564	0.2547	0.5883	**0.2399**	0.2418
	MAE	0.1916	0.3193	0.1674	0.3522	0.1563	0.1722
	Error$_{pos}$	0.2824	0.2588	0.2373	0.5895	0.2785	0.2626
	Error$_{neg}$	0.1625	0.3288	0.1295	0.2335	0.0976	0.134
cpu_pref	RMSE	**0.0332**	0.3907	0.0379	0.0994	0.0856	0.0379
	MAE	0.0231	0.3889	0.0298	0.0936	0.0705	0.0298
	Error$_{pos}$	0.0232	-	0.0238	-	-	0.0238
	Error$_{neg}$	0.023	0.3889	0.0395	0.0936	0.0705	0.0394
hungary chickenpox	RMSE	0.4892	0.7157	**0.487**	0.5463	0.486	**0.487**
	MAE	0.3607	0.6105	0.3484	0.408	0.3535	0.3492
	Error$_{pos}$	0.4542	0.519	0.4442	0.5324	0.4482	0.4517
	Error$_{neg}$	0.3112	0.627	0.2861	0.3421	0.299	0.2852
istanbul stock exchange data	RMSE	0.5783	0.5798	0.5709	0.7203	**0.5468**	0.5644
	MAE	0.4502	0.4513	0.4443	0.5349	0.4281	0.4512
	Error$_{pos}$	0.4889	0.4835	0.4838	0.4758	0.4543	0.5079
	Error$_{neg}$	0.4054	0.4109	0.3967	0.5837	0.3928	0.3378
mcs_ds_edited_iter_shuffled	RMSE	0.1014	0.6715	0.0978	0.4622	0.0987	**0.0812**
	MAE	0.0855	0.5937	0.0796	0.4453	0.0808	0.0623
	Error$_{pos}$	0.0781	0.0189	0.0878	-	0.074	0.0817
	Error$_{neg}$	0.0974	0.6224	0.0727	0.4453	0.0864	0.0342
qsar_aquatic toxicity	RMSE	0.679	0.6224	0.5708	0.6197	**0.3418**	0.5348
	MAE	0.5889	0.4765	0.5055	0.3969	0.2175	0.2557
	Error$_{pos}$	1.1452	1.0737	0.8531	0.6731	0.2986	0.2741
	Error$_{neg}$	0.5135	0.3737	0.45	0.1206	0.0596	0.0731
Daily_Demand_Forecasting_Orders	RMSE	0.0426	0.4943	0.0493	0.3011	**0.0424**	**0.0424**
	MAE	0.0361	0.4919	0.0437	0.2994	0.0349	0.0345
	Error$_{pos}$	0.0307	-	0.0535	-	0.0271	0.0259
	Error$_{neg}$	0.047	0.4919	0.0367	0.2994	0.0505	0.0518
bodyfat	RMSE	0.0506	1.1625	0.0567	0.3515	0.0577	**0.037**
	MAE	0.0381	1.0902	0.047	0.3448	0.0469	0.0282
	Error$_{pos}$	0.0559	-	0.0537	-	0.0493	0.0265
	Error$_{neg}$	0.0284	1.0902	0.0422	0.3448	0.0443	0.0305
machine	RMSE	**0.0329**	0.3924	0.0382	0.1085	0.0944	0.0382
	MAE	0.0223	0.3893	0.0299	0.104	0.0875	0.0301
	Error$_{pos}$	0.0223	-	0.0242	-	0.0241	0.0254
	Error$_{neg}$	0.0222	0.3893	0.0383	0.104	0.0891	0.037
slump_test	RMSE	0.0308	0.2741	0.0329	0.3156	0.0443	**0.0251**
	MAE	0.0285	0.2261	0.0286	0.3146	0.0372	0.0209
	Error$_{pos}$	0.0309	0.1493	0.0321	-	0.0436	0.0238
	Error$_{neg}$	0.027	0.2675	0.022	0.3146	0.0116	0.0197
Airfoil_Self_Noise	RMSE	0.2286	0.2967	0.2268	0.3283	0.14	**0.0864**
	MAE	0.1873	0.2702	0.1764	0.2374	0.0995	0.0505
	Error$_{pos}$	0.2617	0.2283	0.2531	0.3563	0.1235	0.0699
	Error$_{neg}$	0.1551	0.279	0.1334	0.1334	0.0432	0.0088
Delta_Ailerons	RMSE	0.3653	0.507	0.3637	**0.1217**	0.1884	0.1881
	MAE	0.3026	0.4186	0.3003	0.0953	0.1368	0.144
	Error$_{pos}$	0.2924	0.1732	0.3115	0.0831	0.0984	0.1378
	Error$_{neg}$	0.311	0.4846	0.2895	0.106	0.1664	0.1525
Forest_Fires	RMSE	0.1487	0.4088	**0.1183**	0.4046	**0.1183**	**0.1183**
	MAE	0.0994	0.394	0.0935	0.2079	0.0937	0.0936
	Error$_{pos}$	0.0322	0.6846	0.0653	0.583	0.0658	0.0655
	Error$_{neg}$	0.1109	0.3883	0.1654	0.1164	0.1649	0.1653
Yacht_Hydrodynamics	RMSE	1.0056	1.1753	0.5702	1.1828	**0.4899**	0.5998
	MAE	0.9105	0.8238	0.532	1.0161	0.3823	0.4612
	Error$_{pos}$	1.0019	0.8958	0.5362	1.2906	0.282	0.3786
	Error$_{neg}$	0.8248	0.2683	0.5309	0.6043	0.4474	0.5801
2D_Planes	RMSE	*	2.163	1.4336	**0.1415**	1.5517	1.1251
	MAE	*	1.9962	1.2799	0.1249	1.3526	0.9513
	Error$_{pos}$	*	1.9949	1.0785	0.1277	1.6312	1.003
	Error$_{neg}$	*	1.9973	1.4062	0.1223	0.9783	0.8978
Ailerons	RMSE	0.1524	0.1587	0.1195	**0.0049**	0.0537	0.1365
	MAE	0.1461	0.1513	0.107	0.0026	0.0425	0.1094
	Error$_{pos}$	0.1316	0.1714	0.0963	0.0035	0.0447	0.1302
	Error$_{neg}$	0.1654	0.1255	0.1219	0.0021	0.0415	0.0426

Table 5.5: (Continued) RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the benchmark UCI datasets for the proposed HE-LSSVR model and the baseline models.

Dataset	Metric	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
Parkinsons_Telemonitoring	RMSE	0.0349	0.4306	0.0449	**0.0212**	0.0432	0.0385
	MAE	0.0291	0.4283	0.0379	0.0176	0.0368	0.0321
	Error$_{pos}$	0.0206	-	0.0398	0.0049	0.0342	0.033
	Error$_{neg}$	0.0316	0.4283	0.0362	0.0196	0.0392	0.0313
Pole_Telecomm	RMSE	1.0588	1.1328	0.8054	0.7058	0.7661	**0.5421**
	MAE	0.9695	0.9034	0.7546	0.612	0.695	0.4284
	Error$_{pos}$	1.378	1.6409	0.8935	0.8004	0.5558	0.4653
	Error$_{neg}$	0.7773	0.5044	0.6924	0.4932	0.7403	0.408
Average RMSE		0.3113	0.6629	0.3266	0.3901	0.30	0.2736
Average Rank		3.53	5.61	3.03	4.17	2.53	2
Average Time		34.7984	11.7617	0.0039	0.001	0.0741	0.0049

† represents the proposed model.
The boldface denotes the best model corresponding to each dataset.
* denotes that the Matlab program encounters an "out of memory" error.

address this, Iman and Davenport [62] introduced a more robust statistic:

$$F_F = \frac{(D-1)\chi_F^2}{D(p-1)-\chi_F^2}, \tag{5.17}$$

which follows F distribution with $((p-1),(p-1)(D-1))$ d.f.. Table 5.7 presents the results of the Friedman test on 18 benchmark UCI datasets. For $p = 6$ and $D = 18$, we obtain $\chi_F^2 = 38.0762$ and $F_F = 12.4662$. Also, F_F is distributed with $(5, 85)$ d.f.. From the statistical F-distribution table, at 5% level of significance, the value of $F_{(5,85)} = 4.55$. Since $F_F > 4.55$, we reject the null hypothesis. Hence, substantial differences exist among the models. Next, we conduct the Nemenyi post hoc test [25] to compare the models pairwise. The disparity in performance between the two models is considered significant if the corresponding average ranks display a noticeable difference surpassing a specific threshold value known as the critical difference ($C.D.$). When the discrepancy in mean ranks between the two models exceeds the $C.D.$, the model with the lower mean rank is deemed statistically superior because rank 1 denotes the best performance. The computation of $C.D.$ follows

the following formula:

$$C.D. = q_\alpha \sqrt{\frac{p(p+1)}{6D}}. \tag{5.18}$$

Here, q_α is derived from the studentized range statistic divided by $\sqrt{2}$. This value is referred to as the critical value for the two-tailed Nemenyi test. From the distribution table, at 5% level of significance, the value of q_α is 2.569. Thus, after calculation, we obtain $C.D. = 1.78$. Table 5.8 presents the results of the Nemenyi post hoc test on UCI datasets. The average rank differences of the proposed HE-LSSVR from SVR, LS-SVR, BLSSVR, RVFL, and BXSVR are 1.53, 3.61, 1.03, 2.17, and 0.53 respectively. Notably, the disparities in the average ranks for LS-SVR and RVFL exceed the $C.D.$; thus, according to the Nemenyi post hoc test, the proposed HE-LSSVR is significantly better than LS-SVR and RVFL. It is noteworthy that the proposed HE-LSSVR model does not exhibit a statistically significant difference from the SVR, BLSSVR, and BXSVR models; however, it outperformed the SVR, BLSSVR, and BXSVR models on 15, 14, and 13 datasets, respectively, out of the 18 datasets (see Table 5.5). Accordingly, the evidence supports a statistically significant advantage over LS-SVR and RVFL and competitive performance against SVR, BLSSVR, and BXSVR; it does not support an unqualified claim of superiority over all baselines.

To show the efficacy of the proposed HE-LSSVR model in terms of computation, we also evaluated the training time of the models. Table 5.6 illustrates the training times for both baseline models and the proposed HE-LSSVR models across different datasets. The 2D_Planes dataset was left out because it had a memory error during training for the SVR model.

The average training times for SVR, LS-SVR, BLSSVR, RVFL, BXSVR, and HE-LSSVR are 34.7984s, 11.7617s, 0.0039s, 0.001s, 0.0741 and 0.0049s, respectively. These

Table 5.6: Training time in seconds of the proposed HE-LSSVR model and baseline models on benchmark UCI datasets.

Dataset	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
2014_2015 CSM dataset	0.0119	0.0042	0.0037	0.0004	0.0683	0.0027
cpu_pref	0.0117	0.0046	0.0033	0.0001	0.0583	0.0037
hungary chickenpox	0.0925	0.4889	0.0045	0.0003	0.0752	0.003
istanbul stock exchange data	0.0657	0.0537	0.0034	0.0004	0.0723	0.0031
mcs_ds_edited_iter_shuffled	0.004	0.002	0.0035	0.0001	0.0726	0.0032
qsar_aquatic toxicity	0.0839	0.0362	0.0037	0.0002	0.0578	0.0029
Daily_Demand_Forecasting_Orders	0.0029	0.001	0.0035	0.0001	0.0737	0.0031
bodyfat	0.0136	0.0082	0.0018	0.0002	0.0769	0.0044
machine	0.0135	0.0049	0.0041	0.0001	0.0684	0.0039
slump_test	0.0069	0.0028	0.0041	0.0001	0.0713	0.0033
Airfoil_Self_Noise	0.6113	0.3825	0.004	0.0002	0.0803	0.0032
Delta_Ailerons	66.606	24.4937	0.0053	0.001	0.0817	0.0074
Forest_Fires	0.0616	0.0371	0.0037	0.0003	0.0863	0.0028
Yacht_Hydrodynamics	0.0177	0.0142	0.0035	0.0001	0.0731	0.0112
Ailerons	152.7565	71.5022	0.006	0.0053	0.0831	0.0043
Parkinsons_Telemonitoring	24.9768	7.6544	0.0041	0.0016	0.0825	0.0103
Pole_Telecomm	346.2366	95.2581	0.0045	0.0068	0.0788	0.0115
Average Time	34.7984	11.7617	0.0039	0.001	0.0741	0.0049

† represents the proposed model.

results manifest the exceptional efficiency of the proposed HE-LSSVR, with significantly lower training times compared to SVR and LS-SVR. However, the proposed HE-LSSVR takes a slightly longer time to train than the BLSSVR and RVFL, although the difference in training time is minor. It is crucial to note that the comparative evaluation of RMSE and training time demonstrates the overall performance of the models. In this regard, the proposed HE-LSSVR model stands out as it strikes a balance between RMSE and training time, making it superior to the baseline models.

Table 5.7: Illustrate the results of the Friedman test on 18 benchmark UCI datasets.

p	D	χ_F^2	F_F	$F((p-1),(p-1)(D-1))$	Significant difference (As per Friedman test)
6	18	38.0762	12.4662	4.55	Yes

Table 5.8: Difference in the RMSE rankings of the proposed HE-LSSVR model against baseline models on UCI datasets.

Model	Average rank	Rank difference	Significant difference (As per Nemenyi post hoc test)
SVR [5]	3.53	1.53	No
LS-SVR [8]	5.61	3.61	Yes
BLSSVR [47]	3.03	1.03	No
RVFL [16]	4.17	2.17	Yes
BXSVR [49]	2.53	0.53	No
HE-LSSVR†	2	-	N/A

† represents the proposed model.

5.4.3 Experiments on Synthetic Datasets

To further analyze and compare the proposed HE-LSSVR with baseline models, in this subsection we construct synthetic datasets with varying levels of noise. A detailed description of the synthetic datasets used for evaluation is provided in Table 5.9. We have generated five artificial datasets, each of which is duplicated with three different types of noise (Gaussian noise, uniform noise, and student random variable noise), resulting in a total of 15 artificial datasets, each containing 500 samples. Table 5.10 presents the RMSE values for both the proposed HE-LSSVR model and the baseline models on synthetic datasets. Across the 15 synthetic datasets examined, the proposed HE-LSSVR model outperforms the baseline models on 14 datasets, demonstrating its superior performance in the majority of cases. This observation significantly underscores the robustness and effectiveness of the proposed HE-LSSVR model in comparison to the baseline models.

Additionally, to rigorously assess the robustness of HE-LSSVR against outliers, we systematically introduced outliers into the synthetic datasets. Outliers were added by modifying 10% of the total samples in each dataset. For each selected sample, a feature was randomly chosen, and its value was altered by multiplying it by an outlier factor of 10. This systematic introduction of outliers enabled us to thoroughly evaluate the robustness and stability of the HE-LSSVR model in the presence of extreme deviations. Table 5.11 presents the

results of the proposed HE-LSSVR and the baseline models on the outlier-added synthetic datasets. The experimental results show that HE-LSSVR consistently performs better on these outlier-present datasets. Specifically, HE-LSSVR achieves the best performance on 7 out of the 15 synthetic datasets, showcasing its superiority. Moreover, on the remaining 8 datasets, the proposed model secures the second-best position, further underscoring its stability and competitive performance against baseline models. These findings validate the proposed HE-LSSVR as a reliable and robust choice for scenarios involving significant data contamination and outliers.

Table 5.9: Description of synthetic datasets.

Name	Function expression	ϱ	Variable domain
Function 1	$y_i = \sin x_i + \varrho_i$		$x_i \in [0, 2\pi]$
Function 2	$y_i = \frac{\sin(3x_i)}{3x_i} + \varrho_i$		$x_i \in [-4, 4]$
Function 3	$y_i = \sin x_i \cos x_i^2 + \varrho_i$	$\varrho_i \sim N\left(0, 0.2^2\right)$, $U[-0.2, 0.2]$, $T(10)$	$x_i \in [0, 2\pi]$
Function 4	$y_i = x_i \cos x_i + \varrho_i$		$x_i \in [-4, 4]$
Function 5	$y_i = (1 - x_i + 2x_i^2) \exp\left(\frac{-x_i^2}{2}\right) + \varrho_i$		$x_i \in [-4, 4]$

Here, $N\left(0, d^2\right)$ represents the Gaussian random variable with zero mean and standard deviation d.
$U[-a, a]$ represents the uniformly random variable in $[-a, a]$.
$T(c)$ represents the student random variable with c degree of freedom.

Table 5.10: RMSE values on the synthetic datasets for the proposed HE-LSSVR model against baseline models.

Dataset	Metric	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
Type 1: Gaussian noise with 0 mean and 0.2 standard deviation							
Function 1	RMSE	0.12381	0.70565	0.12153	0.06004	2.37E-02	**9.84E-06**
Function 2	RMSE	0.0005	0.64655	0.00459	0.19324	9.00E-03	**1.93E-05**
Function 3	RMSE	0.09197	0.70557	0.12153	0.04986	1.34E-02	**7.59E-06**
Function 4	RMSE	0.00127	0.64479	0.03179	0.14268	3.80E-03	**8.52E-05**
Function 5	RMSE	0.00067	0.6585	0.0296	0.23571	3.40E-03	**7.34E-05**
Type 2: Uniform noise over the interval [-0.2, 0.2]							
Function 1	RMSE	0.07732	0.70559	0.12114	0.04248	2.37E-02	**9.84E-06**
Function 2	RMSE	0.00023	0.64655	0.00459	0.22081	1.77E-02	**1.93E-05**
Function 3	RMSE	0.0849	0.70555	0.12153	0.0558	2.37E-02	**7.59E-06**
Function 4	RMSE	0.00101	0.64162	0.01218	0.17841	1.22E-02	**1.58E-05**
Function 5	RMSE	0.00061	0.66018	0.0296	0.31262	8.00E-03	**7.27E-05**
Type 3: Student random variable noise with 10 degree of freedom							
Function 1	RMSE	0.10258	0.7056	0.12153	0.04949	2.37E-02	**4.10E-06**
Function 2	RMSE	0.00103	0.64677	0.00459	0.20332	6.20E-03	**5.81E-05**
Function 3	RMSE	0.00174	0.68897	0.00319	0.15681	5.00E-04	**2.74E-07**
Function 4	RMSE	0.00121	0.64258	0.03179	0.23784	3.70E-03	**1.42E-05**
Function 5	RMSE	**0.00018**	0.6595	0.0296	0.25285	0.006	0.000187

† represents the proposed model.
The boldface denotes the RMSE of the best model corresponding to each dataset.

Table 5.11: RMSE values on the outlier added synthetic datasets for the proposed HE-LSSVR model against baseline models.

Dataset	Metric	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
Type 1: Gaussian noise with 0 mean and 0.2 standard deviation							
Function 1	RMSE	0.1238	0.70565	0.02387	0.061	**0.004**	0.00704
Function 2	RMSE	**0.00001**	0.55561	0.00277	0.19	0.00342	0.00002
Function 3	RMSE	0.09197	0.70557	0.04873	0.0485	**0.0005**	0.00705
Function 4	RMSE	0.00124	0.54392	0.00277	0.145	0.00895	**0.00022**
Function 5	RMSE	0.00037	0.5906	0.00086	0.238	0.00169	**0.00008**
Type 2: Uniform noise over the interval [-0.2, 0.2]							
Function 1	RMSE	0.09904	0.70559	0.05198	0.0415	**0.0005**	0.00705
Function 2	RMSE	**0.00001**	0.55561	0.00277	0.225	0.02246	0.00006
Function 3	RMSE	0.08489	0.70555	0.05198	0.0545	**0.0005**	0.00705
Function 4	RMSE	0.00109	0.54098	0.00073	0.175	0.00055	**0.000001**
Function 5	RMSE	0.00038	0.58818	0.0293	0.315	0.00602	**0.00007**
Type 3: Student random variable noise with 10 degree of freedom							
Function 1	RMSE	0.10258	0.7056	0.04873	0.0505	**0.00844**	0.00904
Function 2	RMSE	0.00006	0.55601	0.00277	0.2	0.01092	**0.00002**
Function 3	RMSE	0.00169	0.66204	0.00648	0.158	**0.00023**	0.00028
Function 4	RMSE	0.00058	0.54283	0.00073	0.24	0.00895	**0.00004**
Function 5	RMSE	0.00037	0.59022	0.0293	0.25	0.01103	**0.00005**

† represents the proposed model.
The boldface denotes the RMSE of the best model corresponding to each dataset.

Table 5.12: RMSE, MAE, Error$_{pos}$, and Error$_{neg}$ values on the time series datasets for the proposed HE-LSSVR model against baseline models.

Dataset (No. of samples, No. of features)	Metric	SVR[5]	LS-SVR[8]	BLSSVR[47]	RVFL[16]	BXSVR [49]	HE-LSSVR†
NNGC1_dataset_D1_V1_003 (430, 5)	RMSE	0.6837	0.7501	0.6192	0.6541	0.6625	**0.5369**
	MAE	0.5544	0.593	0.5043	0.5493	0.5468	0.4444
	Error$_{pos}$	0.5957	0.6757	0.5316	0.4227	0.4911	0.4377
	Error$_{neg}$	0.5168	0.4536	0.4715	0.6404	0.5909	0.4526
NNGC1_dataset_D1_V1_004 (545,5)	RMSE	0.562	0.6041	0.4883	**0.3343**	0.4609	0.4221
	MAE	0.4701	0.5045	0.4047	0.2807	0.3924	0.3459
	Error$_{pos}$	0.2759	0.2825	0.4481	0.2173	0.3769	0.3219
	Error$_{neg}$	0.5699	0.5812	0.3764	0.3203	0.4025	0.3627
NNGC1_dataset_D1_V1_005 (430, 5)	RMSE	0.0307	0.4047	0.0328	0.0364	0.0392	**0.0298**
	MAE	0.0255	0.4028	0.0274	0.0294	0.0319	0.0247
	Error$_{pos}$	0.025	0.4028	0.0222	0.0351	0.0333	0.0252
	Error$_{neg}$	0.0259	-	0.0301	0.0148	0.0303	0.0243
NNGC1_dataset_D1_V1_006 (610, 5)	RMSE	0.1697	0.4305	0.1678	0.2088	0.1715	**0.1566**
	MAE	0.138	0.3945	0.136	0.1647	0.1404	0.128
	Error$_{pos}$	0.1426	0.3967	0.1532	0.1781	0.1318	0.1283
	Error$_{neg}$	0.1337	0.1237	0.1136	0.1512	0.1472	0.1277
NNGC1_dataset_D1_V1_007 (610, 5)	RMSE	0.086	0.4104	0.0877	0.1033	0.0896	**0.0833**
	MAE	0.0673	0.4013	0.069	0.0797	0.0699	0.065
	Error$_{pos}$	0.0679	0.4013	0.0639	0.0844	0.0795	0.0662
	Error$_{neg}$	0.0668	-	0.0724	0.0745	0.0555	0.0638
NNGC1_dataset_D1_V1_008 (540, 5)	RMSE	0.0982	0.4195	0.0998	0.1024	0.0987	**0.0949**
	MAE	0.0773	0.4078	0.0762	0.0769	0.0787	0.075
	Error$_{pos}$	0.0693	0.4078	0.0585	0.0812	0.0772	0.0718
	Error$_{neg}$	0.0868	-	0.0939	0.0712	0.0808	0.0789
NNGC1_dataset_D1_V1_009 (540, 5)	RMSE	0.2481	0.44	0.2368	0.5011	**0.0987**	0.2256
	MAE	0.1938	0.3795	0.1866	0.3936	0.0787	0.1771
	Error$_{pos}$	0.1834	0.4052	0.1905	0.4137	0.0772	0.1746
	Error$_{neg}$	0.2055	0.0972	0.1808	0.3742	0.0808	0.179
NNGC1_dataset_D1_V1_010 (585, 5)	RMSE	0.0908	0.3931	0.0939	0.1224	0.2373	**0.0849**
	MAE	0.071	0.3781	0.0737	0.0963	0.1899	0.0692
	Error$_{pos}$	0.073	0.3781	0.0607	0.1	0.1901	0.0724
	Error$_{neg}$	0.069	-	0.0827	0.0922	0.1898	0.0625
NNGC1_dataset_E1_V1_008 (740, 5)	RMSE	0.2226	0.4569	0.2182	0.1897	**0.0967**	0.2122
	MAE	0.1842	0.4083	0.1773	0.1657	0.076	0.1711
	Error$_{pos}$	0.1644	0.4374	0.147	0.1378	0.062	0.1251
	Error$_{neg}$	0.2103	0.1065	0.2194	0.2127	0.0847	0.2512
NNGC1_dataset_E1_V1_009 (740, 5)	RMSE	0.2132	0.4538	0.207	**0.1953**	0.6625	0.1995
	MAE	0.1778	0.4076	0.1641	0.169	0.5468	0.1677
	Error$_{pos}$	0.1563	0.4267	0.1171	0.1285	0.4911	0.1325
	Error$_{neg}$	0.2068	0.1131	0.2293	0.2561	0.5909	0.2347
NNGC1_dataset_E1_V1_010 (650, 5)	RMSE	0.2676	0.4607	0.2662	0.387	0.4609	**0.2315**
	MAE	0.2139	0.4073	0.2167	0.3155	0.3924	0.1661
	Error$_{pos}$	0.1866	0.4399	0.2139	0.3194	0.3769	0.1329
	Error$_{neg}$	0.2457	0.137	0.2238	0.311	0.4025	0.2003
Average RMSE		0.243	0.4749	0.229	0.2577	0.2287	**0.207**

† represents the proposed model.
The boldface denotes the RMSE of the best model corresponding to each dataset.

5.4.4 Experiments on Time Series Datasets

To validate the superiority of the proposed HE-LSSVR model in real-world applications, we further evaluate it on time series datasets [79]. The experimental results of the proposed HE-LSSVR and the baseline models are presented in Table 5.12. The proposed HE-LSSVR showcases exceptional performance across the time series datasets. It achieves the lowest RMSE on 7 out of 11 datasets, clearly demonstrating its superiority over the baseline models. Even on the remaining 4 datasets, HE-LSSVR performs admirably, securing the second-best RMSE on 3 datasets and ranking third on 1 dataset. This consistent performance highlights the robustness and effectiveness of the HE-LSSVR model in handling diverse time series data, making it a powerful tool for regression tasks. The average RMSE values of the proposed HE-LSSVR and the baseline models (SVR, LS-SVR, BLSSVR, RVFL, and BXSVR) are 0.207, 0.243, 0.4749, 0.229, 0.2287, and 0.2577, respectively. These values represent the overall performance of each model, with lower RMSE values indicating better predictive accuracy. The results strongly reflect the prominence of the proposed HE-LSSVR model compared to the baseline models.

5.4.5 Experiments on Brain Age Dataset

To demonstrate the effectiveness of the proposed HE-LSSVR model in biomedical applications, we conduct an additional evaluation on the brain age dataset. This chapter utilizes T1-weighted MRI scans sourced from the IXI dataset (`http://www.braindevelopment.org/ixi-dataset/`) and OASIS dataset (`http://www.oasis-brains.org/`). The data is categorized into three key groups: (1) cognitively healthy (CH) participants ($N = 876$): mean age $\pm$ standard deviation (SD) is 47.47 ± 19.49 years, with an age range spanning from 18 to 94 years, (2) mild cognitive impairment (MCI) patients ($N = 70$): mean age $\pm$ SD is 76.21 ± 7.18 years, with ages ranging between 62

and 92 years, and (3) Alzheimer's disease (AD) patients ($N = 30$): mean age $\pm$ SD is 78.03 ± 6.91 years, with an age range of 65 to 96 years. The experimental design, including model evaluation and data preprocessing, adheres to the methodology outlined in [111]. To compare the proposed HE-LSSVR model, we selected the three best baseline models (SVR, BLSSVR, and BXSVR) based on their RMSE values on the UCI datasets. The experimental results are presented in Table 5.13. For the CH group, the proposed HE-LSSVR model outperforms all baseline models, achieving the lowest RMSE (3.01) and MAE (2.52). Compared to the best-performing baseline model (BXSVR), HE-LSSVR shows a notable reduction in RMSE from 3.48 to 3.01 and in MAE from 3.05 to 2.52, demonstrating improved predictive accuracy. For the MCI group, HE-LSSVR and BXSVR both achieve the lowest RMSE (3.89), indicating comparable accuracy in age prediction. However, BXSVR slightly outperforms HE-LSSVR in MAE (3.14 vs. 3.29), suggesting that BXSVR has a minor advantage in minimizing absolute errors. The MCI results therefore indicate parity in RMSE and a small MAE advantage for BXSVR, rather than superiority of HE-LSSVR on this subgroup. For the AD group, HE-LSSVR demonstrates improvements over all baseline models in the reported point estimates, achieving the lowest RMSE (4.36) and MAE (3.69). Compared to the best baseline model (BXSVR), HE-LSSVR reduces RMSE from 5.27 to 4.36 and MAE from 4.14 to 3.69. These point estimates suggest potential utility for brain-age prediction but require uncertainty-aware validation before stronger conclusions are drawn.

The retained brain-age results contain one aggregate RMSE and MAE value per model and subgroup, without fold-level predictions, repeated-split results, or bootstrap replicates. Variability measures therefore cannot be recovered retrospectively from Table 5.13. This is particularly important for the MCI ($N = 70$) and AD ($N = 30$) groups, where point estimates can be sensitive to the sampled test subjects. The subgroup comparison should be

regarded as exploratory; future evaluation should report repeated or nested cross-validation estimates with standard deviations or bootstrap confidence intervals.

Table 5.13: RMSE and MAE values of the proposed HE-LSSVR model against baseline models on testing data for CH, MCI, and AD subjects.

Dataset	Metric	SVR[5]	BLSSVR[47]	BXSVR [49]	HE-LSSVR†
CH	RMSE	6.55	3.69	3.48	**3.01**
	MAE	5.16	3.27	3.05	**2.52**
MCI	RMSE	9.78	5.22	**3.89**	**3.89**
	MAE	8.02	4.14	**3.14**	3.29
AD	RMSE	12.77	5.65	5.27	**4.36**
	MAE	10.93	4.83	4.14	**3.69**

† represents the proposed model.
The boldface denotes the best model corresponding to each dataset.

5.5 Chapter Summary

In this chapter, we addressed the limitations of support vector regression (SVR) by introducing the HawkEye loss function, a novel symmetric loss function that is bounded, smooth, and simultaneously possesses an insensitive zone. The comparison with both SVR losses and broader robust regression families supports the more precise claim that, to the best of our knowledge within the surveyed SVR and LSSVR literature, HawkEye is the first loss to combine these three properties with an exact zero-loss interval. Further, we integrated the HawkEye loss function into the least squares framework of SVR and developed a new robust and scalable model called HE-LSSVR. To solve the optimization problem of HE-LSSVR, we employed the adaptive moment estimation (Adam) algorithm, which offers adaptive learning rates and is effective in handling large-scale problems. The experiments on UCI, synthetic, time-series, and brain-age datasets show competitive aggregate performance and statistically significant gains over selected baselines. The narrow or non-significant differences against SVR, BLSSVR, and BXSVR, the non-nested selection

protocol, and the absence of variability estimates for the small MCI and AD subgroups limit stronger generalization or clinical claims.

In the future, there is a potential avenue for delving into the integration of the HawkEye loss function within the domain of deep learning models. The essential characteristics of boundedness and smoothness intrinsic to the HawkEye loss function suggest promising possibilities for its fusion with advanced machine learning and deep learning techniques. This future direction holds the potential to yield substantial advancements and broader applicability across diverse domains.

Chapter 6

Dual-Center RAPID-LSSVM: Radius-Adaptive, Probability, and Class-Imbalance-Driven Weighting for Alzheimer's Diagnosis

This chapter develops the Radius-Adaptive, Probability and Imbalance Driven (RAPID) weighting strategy for learning from noisy, outlier-prone, and class-imbalanced data, with particular emphasis on Alzheimer's disease (AD) diagnosis. Reliable early AD detection is difficult because clinical and biomedical datasets often contain overlapping classes, mislabeled or ambiguous samples, outliers, and skewed class priors. RAPID addresses these challenges through three complementary components: a radius-adaptive proximity weight that preserves the influence of informative boundary samples while reducing the effect of unreliable central or distant samples, a local class-probability term that down-weights ambiguous or potentially mislabeled instances, and an imbalance-ratio term that compensates for class-prior skew. The chapter integrates RAPID into least-squares support vector machine (LSSVM) learning and develops two classifiers: RAPID-LSSVM-I with mean-center

weighting and RAPID-LSSVM-II with median-center weighting. The median-based variant improves robustness under outliers and asymmetric class distributions. Experiments on KEEL, UCI, noisy benchmark datasets, and the ADNI dataset demonstrate the effectiveness of RAPID-LSSVM for robust class-imbalanced learning.

6.1 Introduction

Alzheimer's disease (AD) is a chronic and progressive neurodegenerative condition [119, 120], accounting for approximately 70% of dementia cases [93]. It primarily affects the brain's cognitive functions, leading to disorientation, mood disturbances, and severe cognitive decline [121, 122]. The prevalence of AD is projected to increase dramatically, with estimates suggesting that by 2050, one in every 85 individuals will be affected [104]. Currently, about 6.7 million individuals aged 65 and above are living with AD, making it the sixth-leading cause of mortality in the United States. The socioeconomic impact of AD is profound, encompassing significant healthcare costs, social welfare provisions, and substantial income losses for families, amounting to an estimated 345 billion in the United States in 2023 [123]. Numerous studies have shown that detecting AD at its earliest stages and initiating timely intervention can markedly decelerate disease progression. Consequently, to halt further advancement, treatment should begin as earliest as symptoms become apparent [124]. Recently, there has been a burgeoning interest in leveraging machine learning algorithms for AD diagnosis [125, 126]. Machine learning approaches have shown promising outcomes for AD detection [101, 127]. As a result, this area has become an active field of research over the past decade. For a detailed overview of machine learning models for AD diagnosis, we refer interested readers to [108].

Hyperplane-based classifiers remain attractive for biomedical tabular learning because their objectives are regularized, mathematically transparent, and effective in small-to-

moderate sample regimes [6, 8]. In this chapter, LSSVM is used as the host classifier because it retains the margin-based structure while leading to a linear system rather than a quadratic program. However, its squared-error-type fitting term can still be strongly influenced by noisy samples, outliers, and class-prior skew.

One algorithm-level way to improve robustness is sample weighting. Fuzzy-membership weighting assigns each training point a reliability value so that doubtful samples contribute less to the empirical risk [12, 128]. For class-imbalance learning, this weighting must do more than measure global distance from a class center: it should also preserve informative boundary samples, suppress locally ambiguous observations, and compensate for minority-class under-representation. These requirements are especially important in AD diagnosis, where clinical classes can overlap and minority cases may carry high diagnostic importance.

Despite the extensive research on SVM-based and fuzzy-weighted classifiers for imbalanced learning, several important challenges remain insufficiently addressed. Existing methods often rely on fixed or heuristic membership schemes that do not simultaneously account for data proximity, local class probability, and imbalance severity. Consequently, these approaches may struggle to effectively suppress noisy samples, preserve informative boundary instances, and maintain robustness under skewed class distributions.

Motivated by the above challenges, we introduce a novel weighting scheme called RAPID (Radius-Adaptive, Probability and Imbalance Driven). The proposed mechanism is designed to handle noise, outliers, and class imbalance, which are common in AD diagnosis. At its core, RAPID employs a radius-adaptive proximity rule: for each sample, the weight is determined by its distance to the class-center, assigning maximal weight within a prescribed neighborhood and then decaying smoothly beyond a threshold so that boundary samples retain meaningful influence. To mitigate label noise and ambiguous instances,

RAPID integrates a local class-probability term that down-weights low-confidence samples; to address skewed priors, it incorporates an imbalance-ratio term that elevates minority-class influence. Embedding RAPID into the least-squares SVM framework yields a robust, flexible classifier, RAPID-LSSVM. We instantiate two center definitions, mean and median, resulting in RAPID-LSSVM-I (mean-center) and RAPID-LSSVM-II (median-center), which induce distinct membership profiles. The key contributions of this work can be encapsulated as follows:

1. We introduce **RAPID** (Radius-Adaptive, Probability and Imbalance Driven), a novel, robust, and flexible weighting mechanism that combats noise, outliers, and class imbalance. RAPID assigns maximal weight to samples within a specified neighborhood of the class-center and decays smoothly beyond this radius, preserving the influence of boundary samples. A local class-probability term attenuates likely noisy/ambiguous instances, and an imbalance-ratio term corrects prior skew.

2. We integrate RAPID into least-squares SVM and propose **RAPID-LSSVM**, an efficient classifier for class-imbalance learning (CIL) under noise and outliers. Leveraging the LSSVM formulation, the solution is obtained by solving linear systems rather than a quadratic program.

3. We instantiate two center definitions, *mean* and *median*, yielding **RAPID-LSSVM-I** (mean-center) and **RAPID-LSSVM-II** (median-center). The median-based variant provides robustness to outliers and asymmetric class distributions, inducing distinct membership profiles.

The subsequent sections are organized as follows. Section 6.2 briefly positions existing weighting schemes and their limitations. Section 6.3 introduces the RAPID flexible weighting mechanism and derives the RAPID-LSSVM formulation with its dual-center variants.

Section 6.4 presents experimental results and statistical comparisons on KEEL and UCI benchmarks, with and without injected label noise, as well as on the ADNI dataset. Finally, Section 6.5 concludes the chapter and outlines future directions.

6.2 Related Work

Fuzzy SVM and weighted SVM variants reduce the influence of unreliable samples by assigning a membership value to each training point [12, 13, 129]. Existing schemes can be broadly grouped into center-based methods, hyperplane-based methods, and local-probability or affinity-based methods. Center-based schemes are simple and efficient, but they rely mainly on global distance from a class center. Hyperplane-based schemes use a preliminary separating surface, but their quality depends on the reliability of the initial classifier. Probability or affinity-based schemes incorporate richer local information [54], but they may introduce additional computational overhead.

For this chapter, the essential observation is that a weighting scheme for class-imbalanced biomedical data should jointly address three aspects: geometric proximity, local label consistency, and class-prior imbalance. Existing schemes typically emphasize only one or two of these aspects, which motivates the RAPID construction.

6.3 RAPID Weighting Scheme and RAPID-LSSVM

In this section, we state the chapter-specific limitations of existing weighting schemes and then introduce RAPID (Radius-Adaptive, Probability and Imbalance Driven). By integrating RAPID into the least-squares SVM framework, we obtain noise- and imbalance-robust class-imbalance learning models, collectively referred to as RAPID-LSSVM.

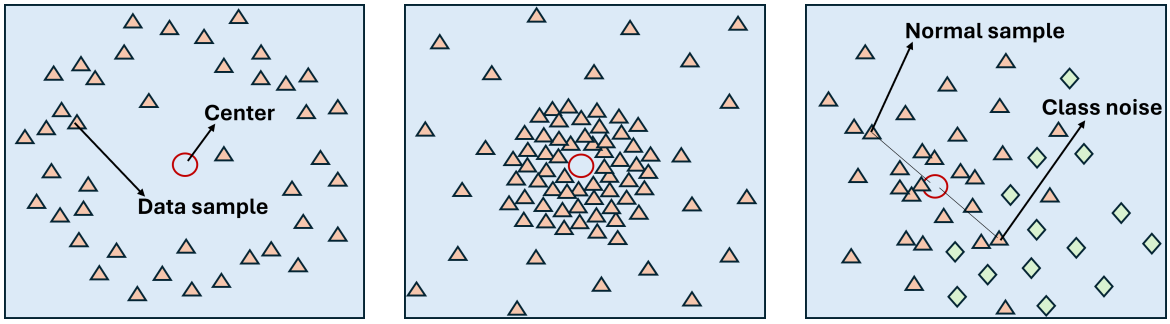
6.3.1 Limitations of Existing Fuzzy Schemes

Existing membership schemes demonstrate resistance against noise and outliers, but the following limitations remain important for the present chapter:

1. Center-based schemes may assign very small weights to informative boundary samples when many points are dispersed away from the class center, as illustrated in Figure 6.1(a).

2. Linear or exponential decay from the class center can underweight samples that are close to the center but not exactly at the center, even when they should be regarded as equally reliable; this is shown in Figure 6.1(b).

3. Distance alone cannot distinguish two samples located at the same radius but embedded in different local neighborhoods. In Figure 6.1(c), a noisy sample and a genuine sample may have comparable distances from the center but should not receive the same reliability.



(a) (b) (c)

Figure 6.1: (a) Depicts a sample distribution where the majority of samples are dispersed farther from the class-center. (b) Illustrates a sample distribution with the majority of samples densely clustered around the class-center. (c) Demonstrates that noise samples are positioned at similar distances from the class-center as the normal samples, highlighting the challenge of distinguishing noise from legitimate data points based solely on their proximity to the center.

4. Center-based memberships are sensitive to how the class center is estimated. The mean can be distorted by outliers or asymmetric distributions, motivating a median-centered alternative [130].

5. Hyperplane-based memberships depend on an initially learned hyperplane. If the initial separator is affected by outliers or class imbalance, the resulting memberships may propagate this error.

6.3.2 Proposed RAPID Weighting Scheme

To address the aforementioned limitations, we propose RAPID, a robust and flexible sample-weighting scheme structured around three components: (1) a *radius-adaptive proximity* mechanism that assigns maximal weight to samples within a neighborhood of the class-center and then decays smoothly beyond a threshold, preserving the influence of boundary samples; (2) a *probability-aware refinement* that down-weights ambiguous or noisy instances using local class-posterior estimates; and (3) an *imbalance-ratio correction* that boosts minority-class influence to counter prior skew. These components are combined to yield the final RAPID membership weights used throughout the RAPID-LSSVM framework.

6.3.2.1 Radius-Adaptive Proximity Mechanism

Classical fuzzy–membership designs typically reduce a sample's weight linearly or exponentially with increasing distance from the class-center, which over-emphasizes points at the center and assigns vanishing weights to boundary samples. In RAPID, we instead adopt a *plateau–then–decay* rule: samples within a neighborhood of the class-center receive the maximal weight, and weights decrease smoothly beyond a tunable threshold so that boundary samples retain influence.

Let $d_i := \|x_i - \mathcal{C}\|$ denote the distance (Euclidean by default) between sample x_i and the class-center $\mathcal{C}$, and let $\mathcal{R}$ be the class radius. For a flexibility parameter $\lambda \geq 1$, the proximity weight is

$$\mu_{\text{flex}}(x_i) \;=\; \begin{cases} 1, & d_i < \dfrac{\mathcal{R}}{\lambda}, \\ \dfrac{\mathcal{R}}{\lambda\, d_i}, & d_i \geq \dfrac{\mathcal{R}}{\lambda}. \end{cases} \tag{6.1}$$

Here, λ regulates the plateau width and the onset of decay: $\lambda = 1$ yields a unit-weight plateau for $d_i < \mathcal{R}$, followed by a $1/d_i$ decay outside the class radius; larger λ values (e.g., $\lambda = 2$) shrink the plateau (to $\mathcal{R}/2$, etc.) and initiate decay earlier. This design is continuous at $d_i = \mathcal{R}/\lambda$, monotonically non-increasing in d_i, bounded by $0 < \mu_{\text{flex}} \leq 1$, and preserves non-negligible weights for boundary-region samples. Figure 6.2a illustrates how varying λ adjusts the plateau width and the decay profile.

6.3.2.2 Probability-Aware Refinement

Distance to the class-center alone cannot distinguish samples that sit at the same radius but lie in differently labeled neighborhoods. To capture local label consistency and suppress likely noise, we estimate a k-NN class posterior around each sample.

Let y_i be the class label of x_i and let $\mathcal{N}_k(x_i) = \{x_{i1}, \ldots, x_{ik}\}$ denote its k nearest neighbors. Define

$$n_i^{\text{own}} \;=\; \sum_{x_j \in \mathcal{N}_k(x_i)} \mathbf{1}\{y_j = y_i\}, \qquad p(x_i) \;=\; \frac{n_i^{\text{own}}}{k} \in [0, 1]. \tag{6.2}$$

Large $p(x_i)$ indicates strong local agreement with the label of x_i; small $p(x_i)$ flags ambiguous or potentially mislabeled points near class boundaries. In RAPID, this probability factor scales the proximity weight (equation 6.1), down-weighting noisy/ambiguous instances while preserving reliable ones. This observation underscores the utility of class probability

in enhancing the robustness of the hyperplane construction process, contributing to a more reliable and accurate classification model.

6.3.2.3 Imbalance-Ratio Correction

To counter skewed class priors, we rescale the weights using the *imbalance ratio (I.R.)*. Without loss of generality, assume the positive class is the minority and the negative class is the majority. Let

$$N_+ = \#\{y_i = +1\}, \qquad N_- = \#\{y_i = -1\}, \qquad \textit{I.R.} = \frac{N_-}{N_+} \ (\geq 1). \tag{6.3}$$

Given the proximity weight $\mu_{\text{flex}}(x_i) \in (0, 1]$ from (6.1), the imbalance-adjusted *proximity* weights are

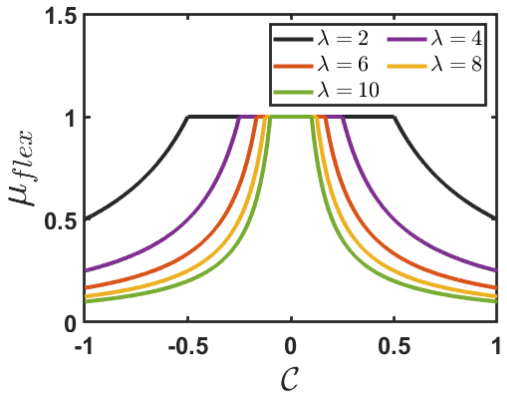
$$\mu_{\text{IR-flex}}(x_i^+) \ = \ \mu_{\text{flex}}(x_i^+) \times \textit{I.R.}, \tag{6.4}$$

$$\mu_{\text{IR-flex}}(x_i^-) \ = \ \mu_{\text{flex}}(x_i^-) \times \frac{1}{\textit{I.R.}}. \tag{6.5}$$

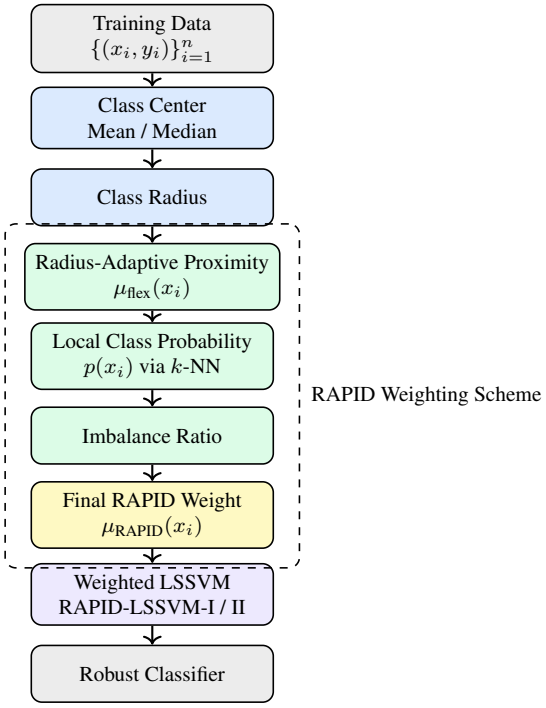
Since $\mu_{\text{flex}}(x_i) \in (0, 1]$, a minority-class sample attains weights in $(0, \textit{I.R.}]$, whereas a majority-class sample lies in $\big(0, 1/\textit{I.R.}\big]$, amplifying informative minority instances while attenuating majority dominance.

Figure 6.2 summarizes the complete RAPID mechanism, including the radius-adaptive proximity component and its integration into the weighted LSSVM pipeline.

Final RAPID Membership: Combining the imbalance-adjusted proximity weights in (6.4)–(6.5) with the class-probability factor in (6.2), the proposed RAPID weighting is as



(a) Radius-adaptive proximity weighting. The flexibility parameter λ controls the plateau width around the class center and the onset of decay, assigning unit weight to central samples and smoothly down-weighting distant samples.



(b) Overall RAPID-LSSVM pipeline illustrating the computation of radius-adaptive proximity, probability-aware refinement, and imbalance-ratio correction, and their integration into the LSSVM framework with a dual-center design.

Figure 6.2: Illustration of the proposed RAPID mechanism. (a) Radius-adaptive proximity weighting showing the effect of the flexibility parameter λ. (b) End-to-end RAPID-LSSVM pipeline integrating proximity, probability, and imbalance-driven weighting into the learning framework.

follows:

$$\mu_{\text{RAPID}}(x_i) = \begin{cases} \mu_{\text{flex}}(x_i) \times \textit{I.R.} \times p(x_i), & x_i \in \text{minority class}, \\ \mu_{\text{flex}}(x_i) \times \dfrac{1}{\textit{I.R.}} \times p(x_i), & x_i \in \text{majority class}. \end{cases} \tag{6.6}$$

Equivalently, $\mu_{\text{RAPID}}(x_i) = \mu_{\text{IR-flex}}(x_i) \times p(x_i)$. For clarity, the complete RAPID weighting pipeline, illustrating the computation of the radius-adaptive proximity, probability-aware refinement, and imbalance-ratio correction, and their integration into the learning framework, is schematically depicted in Fig. 6.2b. Algorithm 6.1 summarizes the step-by-step procedure for computing the RAPID weights.

Algorithm 6.1 RAPID (Radius-Adaptive, Probability and Imbalance Driven) weighting

Input: Training set $\{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{-1, 1\}$; flexibility $\lambda \geq 1$; k-nearest neighbors k; center type $\mathrm{ctr} \in \{\mathrm{mean}, \mathrm{median}\}$.
Output: RAPID membership $\mu_{\mathrm{RAPID}}(x_i)$ for all i.

1. For each class $c \in \{-1, 1\}$, compute class-center $\mathcal{C}_c$ using `ctr` and class radius $\mathcal{R}_c$.
2. For each sample x_i, set $d_i = \|x_i - \mathcal{C}_{y_i}\|$ and compute the proximity weight $\mu_{\text{flex}}(x_i)$ via (6.1) (using $\mathcal{R} = \mathcal{R}_{y_i}$).
3. Estimate the local class probability $p(x_i)$ using k-NN as in (6.2).
4. Compute the imbalance ratio *I.R.* $= N_-/N_+$ (assuming $+1$ is the minority class), where N_+ and N_- are class counts.
5. Obtain *I.R.*-adjusted proximity weights using (6.4)–(6.5): $\mu_{\text{IR-flex}}(x_i^+) = \mu_{\text{flex}}(x_i^+)\,\textit{I.R.}$ and $\mu_{\text{IR-flex}}(x_i^-) = \mu_{\text{flex}}(x_i^-)/\textit{I.R.}$.
6. Compute the final RAPID weight (membership) as $\mu_{\text{RAPID}}(x_i) = \mu_{\text{IR-flex}}(x_i)\, p(x_i)$, as in (6.6).
7. Return $\{\mu_{\text{RAPID}}(x_i)\}_{i=1}^n$.

6.3.3 Computational Complexity

Consider a binary classification task with n training samples in $\mathbb{R}^d$, where n_1 and n_2 denote the numbers of samples in the two classes ($n = n_1 + n_2$). The computational

complexity of the proposed RAPID weighting scheme (equation (6.6)) is decomposed as follows. (i) Radius-adaptive proximity term $\mu_{\text{flex}}(\cdot)$. This step computes the class-center and class-radius for each class and then evaluates the distance of every sample to its corresponding class-center followed by applying the proximity rule in equation (6.1). Computing all distances requires $\mathcal{O}(nd)$ time. The overall complexity of this step is therefore $\mathcal{O}(nd)$. (ii) Probability-aware refinement $p(x_i)$ via k-NN. In our implementation, the class-probability term $p(x_i)$ is computed using a brute-force k-nearest neighbor strategy based on pairwise Euclidean distances. First, the Euclidean distance between every pair of training samples is computed once, resulting in a complete $n \times n$ distance matrix. This step requires $\mathcal{O}(n^2 d)$ time for data in $\mathbb{R}^d$ and $\mathcal{O}(n^2)$ memory. Subsequently, for each sample x_i, the distances to all other samples are sorted in ascending order to identify its nearest neighbors. Sorting n distances for a single sample incurs a cost of $\mathcal{O}(n \log n)$; repeating this process for all n samples results in an overall sorting complexity of $\mathcal{O}(n^2 \log n)$. After sorting, the class-probability $p(x_i)$ is obtained by counting the number of samples sharing the same label as x_i among the first k nearest neighbors, which requires $\mathcal{O}(k)$ operations per sample, or $\mathcal{O}(nk)$ in total. Since $k \ll n$, this term is dominated by the sorting cost. Therefore, the overall time complexity of the probability-aware refinement step, as implemented, is $\mathcal{O}(n^2 d) \; + \; \mathcal{O}(n^2 \log n) \; + \; \mathcal{O}(nk)$, with a memory complexity of $\mathcal{O}(n^2)$ due to the storage of the full pairwise distance matrix. (iii) Imbalance-ratio correction ($I.R.$). The imbalance ratio is computed using class sample counts, which requires $\mathcal{O}(n)$ time. Applying the corresponding scaling factors to all samples incurs an additional $\mathcal{O}(n)$ cost. Combining components (i)-(iii), the overall training-time complexity of the RAPID weight computation is dominated by the probability-aware refinement step. In particular, the brute-force k-NN search requires computing pairwise distances and sorting operations, leading to a complexity of $\mathcal{O}(n^2 d) + \mathcal{O}(n^2 \log n)$. The remaining operations, including proximity

evaluation and imbalance-ratio scaling, incur additional costs of $\mathcal{O}(nd)$ and $\mathcal{O}(n)$, respectively. Since $n^2 d$ and $n^2 \log n$ dominate for typical dataset sizes, the overall complexity of the RAPID weighting computation is governed by the quadratic scaling of the k-NN refinement step. Importantly, this computation is performed only once during training to obtain fixed membership weights M_i and therefore does not affect the prediction-time complexity of RAPID-LSSVM.

6.3.4 Proposed RAPID-LSSVM

By integrating the proposed RAPID weighting scheme into the LSSVM framework, we obtain a robust classifier for class-imbalance learning, termed RAPID-LSSVM. The objective function of the proposed RAPID-LSSVM is formally defined as follows:

$$\begin{aligned} \min_{w,b} \ & \frac{1}{2}\|w\|^2 + \frac{1}{2}C\sum_{i=1}^{n}\mathcal{M}_i\xi_i^2 \\ \text{s.t.} \ & y_i\left(w^\top\phi\left(x_i\right)+b\right) = 1-\xi_i, \quad i=1,2,\ldots,n, \end{aligned} \tag{6.7}$$

where $C > 0$ is the regularization term and $\mathcal{M}_i$ denotes the RAPID membership (i.e., $\mathcal{M}_i = \mu_{\text{RAPID}}(x_i)$ from (6.6)) As the dimensionality of w becomes infinite, solving the primal problem (6.7) directly is not feasible. Consequently, we proceed by formulating the corresponding Lagrangian:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \frac{1}{2}C\sum_{i=1}^{n}\mathcal{M}_i\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left\{y_i\left[w^\top\phi\left(x_i\right)+b\right]-1+\xi_i\right\}, \tag{6.8}$$

where α_i's are the Lagrangian multipliers. The optimal conditions are given as follows:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i), \tag{6.9}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{6.10}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \rightarrow \alpha_i = C\mathcal{M}_i \xi_i, \quad i = 1, \ldots, n, \tag{6.11}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow y_i \left[w^\top \phi(x_i) + b \right] - 1 + \xi_i = 0, \quad i = 1, \ldots, n. \tag{6.12}$$

After the elimination of the variables w and ξ, a system of linear equations can be obtained as follows:

$$\begin{bmatrix} 0 & -Y^\top \\ Y & \Omega + (\mathcal{M}C)^{-1} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_n \end{bmatrix}, \tag{6.13}$$

where $1_n = [1, 1, \ldots, 1] \in R^n$, $(\mathcal{M}C)^{-1} = \text{diag}\left\{ \frac{1}{\mathcal{M}_1 C}, \frac{1}{\mathcal{M}_2 C}, \ldots, \frac{1}{\mathcal{M}_n C} \right\}$, $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$, $Y = [y_1, y_2, \ldots, y_n]$, and $\Omega_{ij} = y_i y_j \mathcal{K}(x_i, x_j)$, with $\mathcal{K}$ the kernel function. After obtaining the optimal α and b, the following decision function can be utilized to predict the label of a new sample $\hat{x}$.

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i \mathcal{K}(x_i, \hat{x}) + b \right). \tag{6.14}$$

Dual approach to class-center determination:

Determining the class-center is pivotal in the proposed RAPID weighting scheme. Although the arithmetic mean is commonly used, it is sensitive to outliers and skewed distributions, which can distort the estimated center [130]. In contrast, the median is a robust location estimator that remains stable under extreme values and asymmetry [131]. Accordingly, RAPID supports both mean- and median-based centers, enabling reliable weighting in noisy, non-symmetric settings typical of AD diagnosis.

Key remarks:

1) Employing two center estimators yields two RAPID-LSSVM variants, enhancing adaptability across datasets and noise regimes.

2) The mean- and median-centered models are denoted RAPID-LSSVM-I and RAPID-LSSVM-II, respectively; the latter is particularly robust to outliers and skewed class distributions.

6.4 Experimental Results

To assess the effectiveness of the proposed models, i.e., RAPID-LSSVM-I and RAPID-LSSVM-II, we evaluate their performance against baseline models including SVM [6], LSSVM [8], FSVM [12], FSVM-CIL-Lin [13], FSVM-CIL-Exp [13], ACFSVM [54], CGFTSVM-ID [132], RVFL [16], GB-RVFL [57], and dRVFL [20]. We utilized publicly available UCI [80] and KEEL [79] benchmark datasets. Moreover, we deploy the proposed models on the AD dataset, accessible through the Alzheimer's Disease Neuroimaging Initiative (ADNI) ($adni.loni.usc.edu$).

6.4.1 Experimental Setup

The experimental configuration includes a PC equipped with an Intel(R) Xeon(R) Gold 6226R CPU running at a speed of 2.90GHz and featuring 128 GB of RAM. The system operates on the Windows 11 platform and executes all experiments using MATLAB R2023a. All datasets are randomly split into training and testing subsets, with a ratio of $70 : 30$, respectively. The Gaussian kernel function is employed in all experiments to map the input samples into a higher-dimensional space. The Gaussian kernel is defined as: $\mathcal{K}(x_1, x_2) = e^{-\frac{\|x_1 - x_2\|^2}{\sigma^2}}$, where σ is the kernel parameter. For each model, the regularization parameter C and kernel parameter σ are selected from the set $\{10^i \mid i = -5, -4, \ldots, 4, 5\}$

and $\{2^i \mid i = -5, -4, \ldots, 4, 5\}$, respectively. For CGFTSVM-ID [132], the gbell membership function parameter takes the range $[1/2, 5/8, 3/4, 7/8, 1]*$(radius of positive/negative class). For FSVM-CIL-Lin, positive constant δ is fixed to 10^{-6}. For FSVM-CIL-Exp, hyperparameter γ is selected from the range $[0.1 : 0.1 : 1]$. The membership hyperparameters for ACFSVM are adopted the same as in [54]. For the proposed models, the flexible parameter λ and the k-nearest neighborhood parameter are both selected from the range $[1 : 1 : 10]$. The performance of the models significantly depends on the selection of parameters [103, 133]. For the RVFL and GB-RVFL models, the regularization hyperparameter C is selected from $\{10^i \mid i = -5, -4, \ldots, 4, 5\}$ and the number of hidden nodes is selected within the range $3 : 20 : 203$. For the dRVFL model, the regularization parameter C is varied over the same range as that used for the RVFL model, the number of hidden layers is chosen from $\{1, 2, \ldots, 10\}$, the number of hidden nodes per layer is selected from $\{256, 512, 1024\}$, and the scaling factor for the randomly generated weights is varied over the range $\{2^i \mid i = -5, -4, \ldots, 4, 5\}$. In order to adjust them, we employ a five-fold cross-validation technique along with a grid search approach. The generalization performance of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models have been evaluated by comparing it with baseline models across various metrics including accuracy (ACC), and f-measure. The same 70:30 partition, preprocessing, and model-specific grid-search framework are used for the proposed methods and their baselines, yielding a controlled comparison on that partition. A single train–test split nevertheless provides a split-dependent estimate, and tuning dataset-dependent parameters such as λ and k without an independent outer evaluation can introduce selection bias [78]. The reported results should therefore be interpreted as evidence under the stated partition; repeated stratified or nested cross-validation is needed to quantify variability across splits and to support stronger

out-of-sample claims. Mathematically,

$$Accuracy\ (ACC) = \frac{\mathcal{TP} + \mathcal{TN}}{\mathcal{TP} + \mathcal{TN} + \mathcal{FP} + \mathcal{FN}}, \tag{6.15}$$

$$F\text{-}measure = \frac{2 \cdot \mathcal{TP}}{2 \cdot \mathcal{TP} + \mathcal{FP} + \mathcal{FN}}, \tag{6.16}$$

where true positive ($\mathcal{TP}$) represents the count of patterns belonging to positive class that are accurately classified, while false negative ($\mathcal{FN}$) signifies the count of patterns belonging to positive class that are inaccurately classified, false positive ($\mathcal{FP}$) denotes the count of patterns belonging to negative class that are inaccurately classified, and true negative ($\mathcal{TN}$) describes the number of data points of negative class that are correctly classified.

6.4.2 Evaluation on KEEL and UCI Datasets

In this subsection, we provide a comprehensive analysis and comparison of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models against various baseline methods on 30 benchmark datasets covering a diverse range of domains. Table 6.1 presents a performance comparison between the proposed RAPID-LSSVM models and baseline models on the KEEL and UCI datasets. For the existing baseline models, SVM, LSSVM, FSVM, FSVM-CIL-Lin, FSVM-CIL-Exp, ACFSVM, CGFTSVM-ID, RVFL, GB-RVFL, and dRVFL, the average accuracies are 89%, 89.96%, 88.82%, 88.87%, 89.14%, 86.52%, 88.77%, 84.7%, 78.67%, and 84.73%, respectively. In contrast, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models achieved average accuracies of 90.02% and 90.30%, respectively. This demonstrates a clear performance improvement, with the proposed RAPID-LSSVM-II model securing the top position and the RAPID-LSSVM-I model achieving the second position in terms of average accuracy. Further, we can deduce that the innovative median

approach used in RAPID-LSSVM-II to determine the class-center proves to be more effective in dealing with real-world data. The superiority of the median approach lies in its robustness to outliers. Unlike the mean, which can be significantly influenced by extreme values, the median provides a more stable central tendency measure, especially in skewed distributions. This characteristic is particularly advantageous in real-world datasets where noise and outliers are prevalent.

Table 6.1: Performance comparison of the proposed RAPID-LSSVM models against the baseline models on KEEL and UCI datasets without label noise.

Dataset	SVM [6]	LSSVM [8]	FSVM [12]	FSVM-CIL-Lin [13]	FSVM-CIL-Exp [13]	ACFSVM [54]	CGFTSVM-ID [132]	RVFL [16]	GB-RVFL [57]	dRVFL [20]	RAPID-LSSVM-I†	RAPID-LSSVM-II†
	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC
breast_cancer_wisc	96.17	98.09	97.61	96.65	96.65	98.09	98.56	72.09	77.91	98.09	98.56	98.56
brwisconsin	95.1	97.06	97.06	95.59	95.59	95.59	96.57	97.56	97.56	95.59	98.04	98.04
chess_krvkp	92.59	94.15	92.59	92.59	93.32	97.29	92.37	90.41	91.45	74.84	96.97	97.29
cleve	80.9	83.15	80.9	79.78	80.9	78.65	79.78	81.11	75.56	79.56	82.02	80.9
credit_approval	87.92	88.41	87.92	87.92	87.92	85.99	85.51	84.62	77.88	81.98	87.92	88.41
echocardiogram	87.18	79.49	89.74	89.74	87.18	76.92	87.18	85	87.5	82.72	82.05	84.62
ecoli-0-1-4-6_vs_5	96.43	98.81	97.62	97.62	97.62	94.05	98.81	98.81	98.81	96.43	98.81	98.81
ecoli-0-3-4-6_vs_5	96.72	98.36	98.36	96.72	96.72	96.72	98.36	98.39	72.13	98.07	98.36	98.36
ecoli-0-3-4-7_vs_5-6	96.1	97.4	96.1	94.81	93.51	92.21	92.4	88.46	62.12	88.05	96.1	94.81
ecoli-0-6-7_vs_3-5	95.45	96.97	95.45	93.94	93.94	92.42	95.45	85.07	82.36	82.97	96.97	96.97
ecoli-0-6-7_vs_5	95.45	98.48	92.42	92.42	93.94	89.39	95.45	93.94	59.09	72.55	98.48	98.48
ecoli3	94	91	90	94	96	90	91	92.08	87.13	89	92	97
hepatitis	86.96	89.13	86.96	86.96	89.13	80.43	84.78	72.34	72.34	86.96	84.78	89.13
horse_colic	82.73	85.45	78.18	79.09	76.36	80	80.64	83.78	72.97	79.09	80	83.64
ilpd_indian_liver	72.41	72.41	72.41	72.41	72.41	74.71	67.82	66.86	63.43	70	74.14	74.14
iono	86.67	91.43	86.67	86.67	89.52	88.57	92.38	88.68	83.96	86.19	92.38	92.38
ionosphere	88.57	91.43	88.57	89.52	91.43	86.67	93.33	82.79	83.96	80	92.38	92.38
monk2	91.67	87.78	91.67	91.67	91.11	67.22	82	76.24	50.83	85.44	86.67	87.78
musk_1	82.39	86.62	82.39	82.39	82.39	74.65	80.73	80.91	75.52	84.08	85.92	90.85
pima	78.7	75.65	79.13	76.96	78.7	76.09	78.26	74.03	79.7	72.26	75.65	77.83
pittsburg_bridges_T_OR_D	93.33	96.67	88.84	93.33	93.33	100	96.67	77.42	83.87	82	100	86.67
spectf	90	91.25	91.25	91.25	91.25	78.75	90	83.95	54.32	88.75	91.25	88.75
statlog_australian_credit	69.57	69.57	69.57	69.57	68.6	69.57	68.12	68.75	62.02	69.57	69.57	69.57
statlog_german_credit	75.33	78.67	75.33	75.33	76	77.33	76	70.33	70	71.67	75.33	75.67
statlog_heart	81.48	85.19	82.72	85.19	85.19	83.95	80.25	81.48	82.72	81.62	83.95	83.95
vehicle2	95.65	94.07	95.65	95.65	95.65	92.49	92.23	90.85	91.73	90.85	95.26	95.26
votes	88.46	96.15	88.46	88.46	88.46	95.38	96.15	90.47	92.37	94.62	95.38	96.15
vowel	100	100	100	100	100	93.58	100	99.33	85.19	89.32	100	100
yeast3	95.51	95.96	93.48	93.26	94.61	93.03	94.61	86.72	92.38	94.11	93.93	94.83
yeast5	96.63	97.08	97.53	96.63	96.63	95.96	97.75	98.43	93.27	95.51	97.75	97.75
Average ACC	89	89.96	88.82	88.87	89.14	86.52	88.77	84.7	78.67	84.73	<u>90.02</u>	**90.30**
Average Rank	6.2	<u>4.13</u>	5.93	6.32	5.83	7.9	6.1	8.27	10.07	9.22	4.33	**3.7**

† represents the proposed models.
Bold and underlined text denote the models with the highest and second-highest average accuracy, respectively.

Relying solely on average accuracy can be limiting, since strong performance on specific datasets may conceal weaker results on others. Consequently, it becomes necessary to rank each model separately for every dataset to provide a more holistic evaluation. According to the ranking scheme outlined in [25], the model showing the poorest performance on a given dataset is assigned the highest rank, whereas the top-performing model receives the lowest rank. When evaluating p models across $\mathcal{D}$ datasets, let $\mathcal{R}_j^i$ denote the rank of the jth model on the ith dataset. The average rank of model j is then calculated as follows: $\mathscr{R}_j = \frac{1}{\mathcal{D}} \sum_{i=1}^{\mathcal{D}} \mathcal{R}_j^i$. The average rank of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II, along with the baseline SVM, LSSVM, FSVM, FSVM-

CIL-Lin, FSVM-CIL-Exp, ACFSVM, CGFTSVM-ID, RVFL, GB-RVFL, and dRVFL, are 4.33, 3.70, 6.20, 4.13, 5.93, 6.32, 5.83, 7.90, 6.10, 8.27, 10.07, and 9.22, respectively. The proposed RAPID-LSSVM-II model achieved the lowest average rank among all methods, while RAPID-LSSVM-I placed third. Since a lower rank indicates superior performance, RAPID-LSSVM-II stands out as the top-performing model. We next perform statistical tests to validate the significance of these findings. Specifically, we use the Friedman test [25] to determine whether there are notable differences among the models. Under the null hypothesis, all models exhibit equivalent performance, as captured by their average ranks. The Friedman test statistic follows a chi-squared (χ_F^2) distribution with $(p-1)$ degrees of freedom (d.f.) and is given by $\chi_F^2 = \frac{12\mathcal{D}}{p(p+1)}\left[\sum_j \mathscr{R}_j^2 - \frac{p(p+1)^2}{4}\right]$. Since the Friedman statistic can be overly conservative, Iman and Davenport [62] proposed a more robust measure: $F_F = \frac{(\mathcal{D}-1)\chi_F^2}{\mathcal{D}(p-1)-\chi_F^2}$ which follows an F-distribution with $(p-1)$ and $(p-1)(\mathcal{D}-1)$ d.f.. For $p = 12$ and $\mathcal{D} = 30$, we obtained $\chi_F^2 = 102.5958$ and $F_F = 13.0836$. From the F-distribution table at 5% level of significance $F_F(11, 319) = 1.8187$. Since $F_F > 1.8187$, we reject the null hypothesis. Hence, substantial differences exist among the models. Next, we apply the Nemenyi post hoc test [25] to analyze pairwise differences among the models. The critical difference ($C.D.$) is defined by $C.D. = q_\alpha\sqrt{\frac{p(p+1)}{6\mathcal{D}}}$, where q_α represents the critical value obtained from the two-tailed Nemenyi test table. Referring to the statistical F-distribution table, a value of $q_\alpha = 3.25$ at the 5% significance level yields a $C.D.$ of 3.0255. The average rank differences between the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models with the baseline SVM, LSSVM, FSVM, FSVM-CIL-Lin, FSVM-CIL-Exp, ACFSVM, CGFTSVM-ID, RVFL, GB-RVFL, and dRVFL models are $(1.87, 2.50)$, $(0.20, 0.43)$, $(1.60, 2.23)$, $(1.99, 2.62)$, $(1.50, 2.13)$, $(3.57, 4.20)$, $(1.77, 2.40)$, $(3.94, 8.27)$, $(5.74, 10.07)$, and $(4.89, 5.52)$ respectively. As per the Nemenyi post hoc test, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II exhibit significant differences compared to the

ACFSVM, RVFL, GB-RVFL, and dRVFL models. Nevertheless, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models outperform the other baseline models with respect to average rank. Taking into account all these findings, we can conclude that the proposed RAPID-LSSVM-I demonstrates a competitive nature, and the proposed RAPID-LSSVM-II showcases superior performance against the existing models.

Table 6.2: Performance comparison of the proposed RAPID-LSSVM models against the baseline models on KEEL and UCI datasets with label noise.

Dataset	Noise	SVM [6]	LSSVM [8]	FSVM [12]	FSVM-CIL-Lin [13]	FSVM-CIL-Exp [13]	ACFSVM [54]	CGFTSVM-ID [132]	RAPID-LSSVM-I†	RAPID-LSSVM-II†
		ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC
chess_krvkp	5%	91.96	93.42	91.96	91.96	92.38	88.62	94.57	94.36	94.89
	10%	88.52	90.61	88.52	88.52	88.41	86.22	88.1	94.47	91.02
	20%	77.24	77.87	77.24	77.24	77.24	77.14	78.29	78.08	77.97
	30%	62.63	69	62.63	62.63	62.63	65.97	68.06	65.87	69
	40%	55.95	55.43	55.95	55.95	55.95	65.97	59.71	55.96	55.43
Average ACC		75.26	77.27	75.26	75.26	75.32	76.78	77.75	77.75	77.66
ecoli-0-1-4-6_vs_5	5%	97.62	96.43	95.24	96.43	95.24	92.86	92.86	98.81	98.81
	10%	95.24	94.05	95.24	94.05	95.24	92.86	97.62	97.62	97.62
	20%	94.05	91.67	94.05	94.05	96.43	92.86	89.29	86.9	85.71
	30%	84.52	94.05	84.52	85.71	84.52	92.86	84.52	92.86	92.86
	40%	80.95	79.76	80.95	79.76	79.76	92.86	58.33	98.81	95.08
Average ACC		90.48	91.19	90	90	90.24	92.86	84.52	95	94.02
ecoli-0-3-4-7_vs_5-6	5%	92.21	94.1	94.81	94.81	93.51	89.61	93.51	97.4	96.1
	10%	96.1	97.4	96.1	96.1	94.81	89.61	97.4	93.51	97.4
	20%	90.91	97.4	87.01	87.01	84.42	67.53	92.21	74.03	76.62
	30%	89.61	89.61	89.61	89.61	93.51	89.61	83.12	92.21	89.61
	40%	71.43	75.32	71.43	68.83	77.92	89.61	75.32	97.4	90.91
Average ACC		88.05	90.77	87.79	87.27	88.83	85.19	88.31	90.91	90.13
votes	5%	88.46	94.62	88.46	88.46	87.69	92.31	95.38	93.85	95.38
	10%	88.46	93.85	88.46	88.46	88.46	95.38	94.62	94.62	94.62
	20%	88.46	90.77	88.46	88.46	88.46	86.15	96.15	90	94.62
	30%	89.23	89.23	89.23	89.23	89.23	82.31	90.77	76.92	90.77
	40%	89.23	86.77	89.23	86.92	88.46	70.77	71.54	88.92	82.31
Average ACC		88.77	91.05	88.77	88.31	88.46	85.38	89.69	88.86	91.54
yeast3	5%	94.83	94.96	94.83	93.26	91.69	91.24	95.51	95.51	95.51
	10%	94.61	94.16	95.06	88.99	93.71	89.21	94.83	95.73	95.73
	20%	92.93	92.36	92.71	80.9	92.58	89.21	94.16	93.03	92.36
	30%	93.03	87.64	92.81	88.31	86.52	89.21	86.74	88.09	88.76
	40%	69.44	73.71	65.39	72.58	75.28	89.21	84.72	84.72	89.21
Average ACC		88.97	88.57	88.16	84.81	87.96	89.62	91.19	91.42	92.31

† represents the proposed models.

6.4.3 Evaluation on KEEL and UCI Datasets with Added Label Noise

KEEL and UCI datasets used in our evaluation reflect real-world circumstances; however, it is essential to acknowledge that data impurities or noise can arise due to various factors. In such scenarios, it is crucial to develop a robust model that can effectively handle such challenges. To showcase the effectiveness of the proposed RAPID-LSSVM models, even in adverse conditions, we deliberately added label noise to selected datasets. We selected 5 diverse datasets for our comparative analysis, namely chess_krvkp, ecoli-0-1-4-6_vs_5, ecoli-0-3-4-7_vs_5-6, votes, and yeast3. To ensure impartiality in evaluating the models, we

selected 2 datasets where the proposed RAPID-LSSVM models do not achieve the highest performance and 1 dataset where the proposed models tie with an existing model. Furthermore, we selected 1 dataset where the proposed RAPID-LSSVM models outperform the baseline models, and 1 dataset where one of the proposed models surpasses the baseline models while the other proposed model ties with an existing model. To carry out a thorough analysis, we added label noise at varying levels of 5%, 10%, 20%, 30%, and 40% to corrupt the labels. The accuracies of all the models for the selected datasets with 5%, 10%, 20%, 30%, and 40% noise are presented in Table 6.2. The proposed models, RAPID-LSSVM-I and RAPID-LSSVM-II, have attained top positions on the chess_krvkp dataset in the absence of noise. Notably, they maintain this leading performance even in the presence of noise. The average accuracies of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II on chess_krvkp with various noise levels are 77.75% and 77.66%, respectively, surpassing the performance of the baseline models. On the ecoli-0-1-4-6_vs_5 dataset, both the proposed models tie with LSSVM at 0% noise level. However, the average accuracies of the proposed models (RAPID-LSSVM-I and RAPID-LSSVM-II) on different noise levels are 95% and 94.02%, respectively, outperforming all the baseline models. On the ecoli-0-3-4-7_vs_5-6 and yeast3 datasets, the proposed models did not secure the top positions at 0% noise level. However, with distinct noise levels, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II secured the first and third positions, respectively, on the ecoli-0-3-4-7_vs_5-6 dataset. Further on the yeast3 dataset, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II achieved the second and first positions, respectively. On the votes dataset, the proposed Flexi-Fuz-LSSVM-II shares the top first position with LSSVM, while the proposed Flexi-Fuz-LSSVM-I shares the second position with ACFSVM at 0% noise level. On different levels of noise, the proposed Flexi-Fuz-LSSVM-II, with an average accuracy of 91.54%, surpasses all the baseline models along with the proposed Flexi-Fuz-LSSVM-I

on votes dataset. The aforementioned findings underscore the significance of the proposed RAPID-LSSVM models as robust models. The primary reason for the success of the proposed models lies in several key factors: (a) the robust membership scheme, which ensures that samples near the class boundary retain significant influence while distinguishing noisy samples from normal ones; (b) the flexible parameter λ and the k-nearest neighborhood parameter k, which enable adaptive adjustments to fine-tune the model and handle different noise levels and data distributions effectively; (c) the use of the median approach in RAPID-LSSVM-II, which enhances robustness by providing a stable reference point that is less susceptible to the effects of outliers and noise compared to the mean approach. These adaptabilities ensure that the models maintain optimal performance across a wide range of scenarios, making them highly versatile for practical applications. The combination of these features allows the proposed models to excel in various noisy and real-world data environments, demonstrating superior accuracy and resilience.

6.4.4 Ablation Study

The proposed RAPID weighting scheme integrates three complementary components: (i) a radius-adaptive proximity mechanism $\mu_{\text{flex}}(\cdot)$ to preserve boundary information while suppressing extreme outliers, (ii) a probability-aware refinement $p(\cdot)$ to down-weight locally ambiguous or potentially mislabeled samples, and (iii) an imbalance-ratio ($I.R.$) correction to counter skewed class distributions. To systematically assess the individual contribution of each component and to verify their necessity, we conduct a comprehensive ablation study.

Specifically, we evaluate the performance of RAPID-LSSVM by removing one component at a time while keeping the remaining components unchanged. The following reduced variants are considered:

- **RAPID-LSSVM** $\setminus \mu_{\textbf{flex}}$: without the radius-adaptive proximity mechanism,

- **RAPID-LSSVM** $\setminus p(x)$: without the probability-aware refinement,
- **RAPID-LSSVM** $\setminus I.R.$: without the imbalance-ratio correction.

These variants are compared against the full RAPID-LSSVM models, namely RAPID-LSSVM-I (mean-center) and RAPID-LSSVM-II (median-center). The ablation experiments are performed on five diverse benchmark datasets, brwisconsin, iono, monk2, pima, and vehicle2. The classification accuracy obtained by each variant is reported in Table 6.3.

Table 6.3: Ablation study analyzing the contribution of individual components of the RAPID weighting scheme. Each variant removes one component at a time, while RAPID-LSSVM-I and RAPID-LSSVM-II correspond to the complete models with mean- and median-based centers, respectively.

Dataset	**RAPID-LSSVM** $\setminus \mu_{\textbf{flex}}$	**RAPID-LSSVM** $\setminus p(x)$	**RAPID-LSSVM** $\setminus I.R.$	**RAPID-LSSVM-I**	**RAPID-LSSVM-II**
brwisconsin	96.57	97.53	97.04	98.04	98.04
iono	91.33	90.38	92.38	92.38	92.38
monk2	86.11	85.22	86.89	86.67	87.78
pima	76.52	76.83	75.39	75.65	77.83
vehicle2	94.86	93.26	92.89	95.26	95.26

From Table 6.3, several important observations can be made. First, removing the radius-adaptive proximity component μ_{flex} consistently degrades performance across datasets, highlighting its role in preserving informative boundary samples while mitigating the influence of central noise and extreme outliers. Second, excluding the probability-aware refinement $p(x)$ leads to noticeable performance drops, particularly on datasets such as iono and monk2, confirming its effectiveness in suppressing ambiguous or mislabeled samples through local label consistency. Third, omitting the imbalance-ratio correction adversely affects performance on imbalanced datasets such as pima, demonstrating the importance of explicitly correcting prior skew. Overall, the full RAPID-LSSVM models consistently outperform all ablated variants, with RAPID-LSSVM-II achieving the best or comparable performance in most cases. This validates that the three components of RAPID are not redundant but instead play complementary roles, and their joint integration is crucial for achieving robustness against noise, outliers, and class imbalance.

6.4.5 Evaluation on ADNI Dataset

AD is a progressive neurodegenerative disorder that adversely impacts memory and cognitive functions. It typically initiates with mild cognitive impairment (MCI), making early detection essential for timely intervention and patient care. While the precise etiology of AD remains not fully understood, accurate and early diagnosis is pivotal in slowing disease progression. In this chapter, we employed the ADNI dataset to evaluate the performance of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models. The ADNI project, initiated by Michael W. Weiner in 2003, aims to assess various neuroimaging modalities, including magnetic resonance imaging (MRI), positron emission tomography (PET), and cognitive biomarkers, for the early and differential diagnosis of AD, particularly in the MCI stage. The feature extraction protocol used in this work is consistent with the approach adopted in [127]. Specifically, we use T1-weighted structural MRI (sMRI) data acquired from the ADNI database. The experimental evaluation is conducted on a total of 813 subjects, comprising 228 cognitively normal (CN), 398 MCI, and 187 AD individuals, all of which are used for both training and testing. The subjects fall within the age range of 60–90 years, with a mean age of 75.83 and a standard deviation of 6.07. The Mini-Mental State Examination (MMSE) scores range from 17 to 30, with a mean of 26.51 and a standard deviation of 2.88. The sMRI scans are acquired using a 1.5T scanner with an MP-RAGE sequence under a 3D acquisition protocol. The imaging parameters include a slice thickness of 1.2 mm, a flip angle of 8°, sagittal acquisition plane, and scanners manufactured by GE Medical Systems. These acquisition settings provide high-resolution structural images suitable for morphometric analysis. Voxel-based morphometry (VBM) preprocessing is performed using Statistical Parametric Mapping (SPM12) (Wellcome Trust Centre for Neuroimaging, University College London, UK). All images are manually reoriented to the anterior commissure and spatially registered using SPM's single-subject T1 template. Unified segmentation

is applied to decompose the images into gray matter (GM), white matter (WM), and cerebrospinal fluid (CSF), followed by template generation and spatial normalization using the DARTEL framework. The normalized images are modulated and smoothed using an 8 mm full-width-at-half-maximum (FWHM) Gaussian kernel, and resampled to a dimension of $121 \times 145 \times 121$ with a voxel size of $1.5\,\text{mm}^3$. Statistically significant brain regions are identified using a two-sample t-test within a general linear model, with age and gender included as covariates. Variations in individual head sizes are controlled by incorporating total intracranial volume (TIV) as a covariate using an ANCOVA-by-subject approach. Statistical significance is determined using a family-wise error (FWE) corrected p-value of 0.05. The resulting voxels of interest are used as masks for feature extraction. Importantly, the DARTEL pipeline was executed separately for the training and testing phases, ensuring realistic evaluation conditions where test images are not available during template construction. In addition to VBM-based features, volumetric and cortical thickness features are extracted using FreeSurfer's `recon-all` pipeline (version 6.0.1). From the successfully processed scans, 23 subcortical volume measures, 34 white-matter volume measures, and 34 cortical thickness features were obtained for each subject. Volumetric features are normalized by TIV, and thickness as well as volumetric measures from both hemispheres are aggregated. Feature dimensionality is subsequently reduced using principal component analysis (PCA), followed by F-score–based feature selection. No data augmentation is applied. Class imbalance inherent in the ADNI cohort is explicitly handled through the proposed RAPID weighting mechanism, enabling robust and reliable learning under realistic clinical conditions. The dataset encompasses three binary classification tasks: CN vs. AD, CN vs. MCI, and MCI vs. AD.

Table 6.4 presents the performance metrics (accuracy and f-measure) of the proposed models in comparison to seven baseline classifiers. The proposed RAPID-LSSVM-II and

RAPID-LSSVM-I models achieve the top two average accuracies of 76.20% and 74.26%, respectively. Among the baseline models, GB-RVFL performs the best with an average accuracy of 73.84%. Compared to GB-RVFL, the proposed RAPID-LSSVM-I and RAPID-LSSVM-II show improvements of approximately 0.42% and 2.36% in accuracy, respectively. In addition to accuracy, f-measure, a harmonic mean of precision and recall that better reflects model performance on imbalanced data, is also analyzed. The proposed RAPID-LSSVM-II achieved the highest average f-measure of 72.74%, followed by RAPID-LSSVM-I with 70.34%, while the best-performing baseline (GB-RVFL) attained an f-measure of 70.17%. This clearly illustrates the superior balance achieved by our proposed models between sensitivity and precision, which is particularly important in medical diagnosis scenarios where false positives and false negatives carry serious consequences.

In summary, the proposed RAPID-LSSVM models, especially RAPID-LSSVM-II, demonstrate strong and consistent performance. These findings validate the models' effectiveness in differentiating between various cognitive conditions and their potential utility in clinical decision support systems for AD.

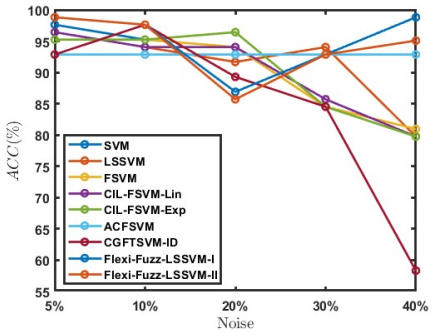
Table 6.4: Performance comparison between the proposed RAPID-LSSVM models and baseline models on the ADNI dataset.

Subjects (Samples × Features)	SVM [6] (ACC, f-measure)	LSSVM [8] (ACC, f-measure)	FSVM [12] (ACC, f-measure)	FSVM-CIL-Lin [13] (ACC, f-measure)	FSVM-CIL-Exp [13] (ACC, f-measure)	ACFSVM [54] (ACC, f-measure)	CGFTSVM-ID [132] (ACC, f-measure)	RVFL [16] (ACC, f-measure)	GB-RVFL [57] (ACC, f-measure)	dRVFL [20] (ACC, f-measure)	RAPID-LSSVM-I† (ACC, f-measure)	RAPID-LSSVM-II† (ACC, f-measure)
CN vs AD (415 × 91)	(79.03, 69.77)	(84.68, 87.27)	(79.03, 69.77)	(79.03, 69.77)	(80.65, 75.51)	(62.1, 65.71)	(57.26, 58.91)	(80, 80.73)	(82, 78.77)	(81.77, 73.55)	(86.29, 82.11)	(87.9, 86.49)
CN vs MCI (626 × 91)	(61.5, 63.27)	(61.57, 65.69)	(62.5, 63.27)	(61.5, 63.27)	(61.5, 63.27)	(45.45, 43.84)	(57.22, 63.64)	(72.87, 80.46)	(70.21, 79.1)	(61.5, 68.34)	(67.91, 76.38)	(68.98, 77.52)
MCI vs AD (585 × 91)	(69.71, 51.06)	(71.43, 45.65)	(69.71, 51.06)	(71.71, 51.06)	(69.57, 51.02)	(68, 68.25)	(66.29, 52.8)	(68.18, 39.13)	(69.32, 52.63)	(68, 57.07)	(68.57, 52.54)	(71.71, 54.21)
Average (ACC, f-measure)	(70.08, 61.37)	(72.56, 66.2)	(70.41, 61.37)	(70.75, 61.37)	(70.57, 63.27)	(58.52, 59.27)	(60.26, 58.45)	(73.68, 68.77)	(73.84, 70.17)	(70.42, 66.32)	(74.26, 70.34)	(**76.20**, **72.74**)

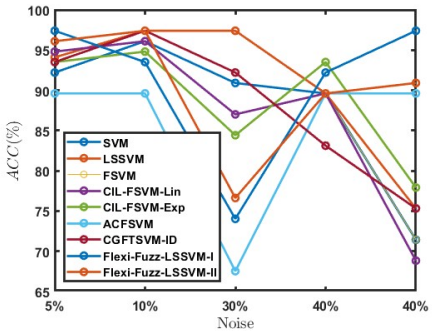
† represents the proposed models.
Bold and underlined text denotes the models with the highest and second-highest result, respectively.

6.4.6 Sensitivity Analysis

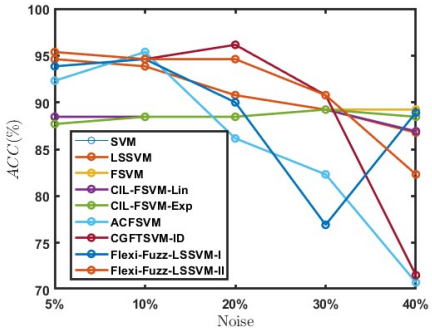
In this subsection, we analyze the sensitivity of the proposed models to noise levels and hyperparameters, and discuss the computational and practical implications of tuning the flexibility parameter λ and the k-nearest neighborhood size k. We first evaluate the resilience of RAPID-LSSVM under varying levels of label noise. Figure 6.3 illustrates the



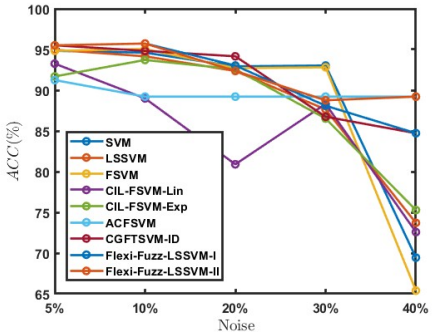
(a) ecoli-0-1-4-6_vs_5



(b) ecoli-0-3-4-7_vs_5-6

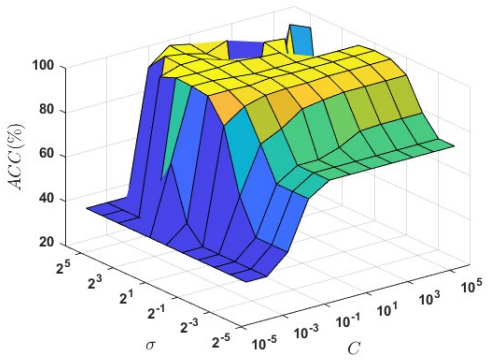


(c) votes

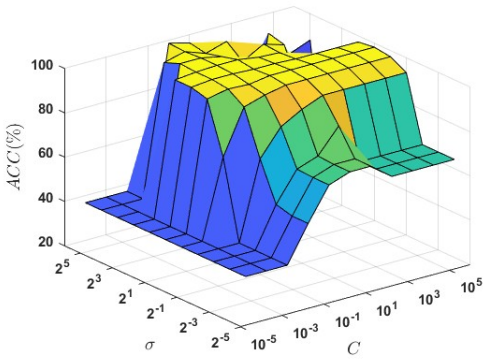


(d) yeast3

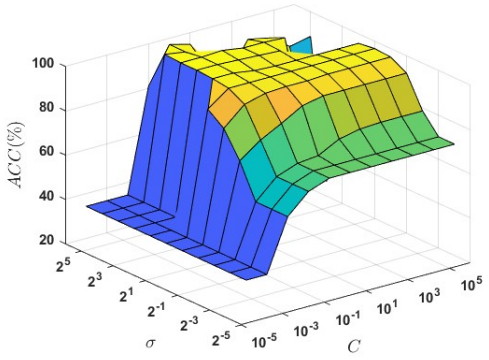
Figure 6.3: Illustrates the impact of various levels of label noise on the performance of the proposed RAPID-LSSVM models compared to the baseline models, demonstrating their robustness against label noise. The datasets used are (a) ecoli-0-1-4-6_vs_5, (b) ecoli-0-3-4-7_vs_5-6, (c) votes, and (d) yeast3.



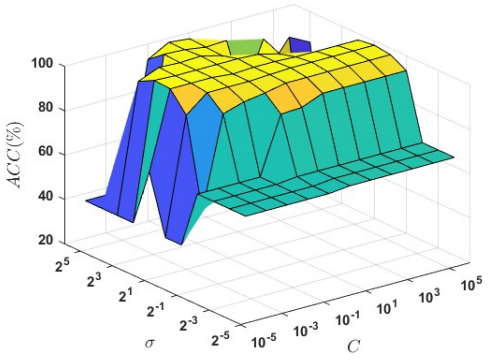
(a) brwisconsin (RAPID-LSSVM-I)



(b) breast_cancer_wisc (RAPID-LSSVM-I)

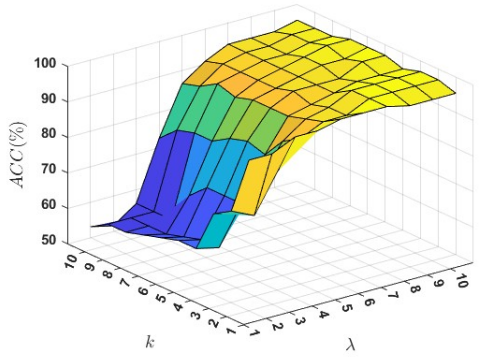


(c) brwisconsin (RAPID-LSSVM-II)

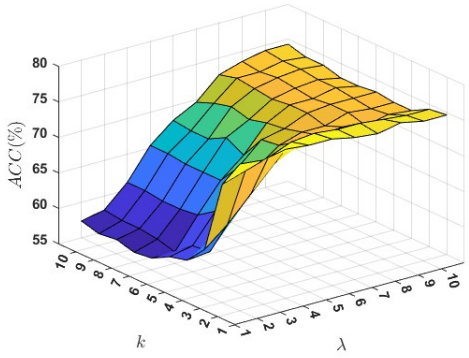


(d) breast_cancer_wisc (RAPID-LSSVM-II)

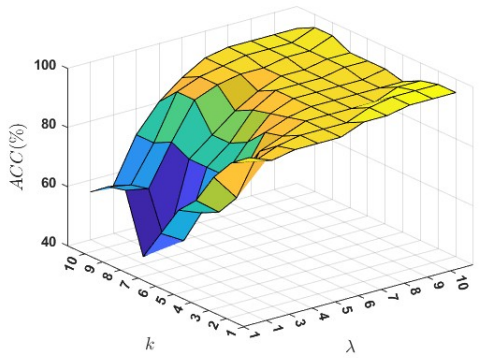
Figure 6.4: Demonstrates the effect of the regularization parameter C and the kernel parameter σ on the performance of the proposed RAPID-LSSVM models. The datasets used for this analysis are (a) brwisconsin with RAPID-LSSVM-I, (b) breast_cancer_wisc with RAPID-LSSVM-I, (c) brwisconsin with RAPID-LSSVM-II, and (d) breast_cancer_wisc with RAPID-LSSVM-II. The surface plots illustrate how changes in C and σ influence the accuracy of the models, highlighting the sensitivity of the RAPID-LSSVM models to these parameters.



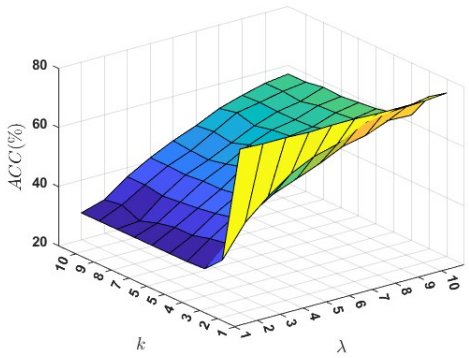
(a) ecoli-0-6-7_vs_3-5 (RAPID-LSSVM-I)



(b) statlog-german-credit (RAPID-LSSVM-I)



(c) ecoli-0-6-7_vs_3-5 (RAPID-LSSVM-II)



(d) statlog-german-credit (RAPID-LSSVM-II)

Figure 6.5: Explores the impact of the flexible parameter λ and the k-nearest neighborhood parameter k on the performance of the proposed RAPID-LSSVM models. The datasets analyzed are (a) ecoli-0-6-7_vs_3-5 with RAPID-LSSVM-I, (b) statlog-german-credit with RAPID-LSSVM-I, (c) ecoli-0-6-7_vs_3-5 with RAPID-LSSVM-II, and (d) statlog-german-credit with RAPID-LSSVM-II. The surface plots illustrate how variations in λ and k parameters affect the models' accuracy, highlighting the sensitivity and influence of these parameters on the performance of the RAPID-LSSVM models.

classification accuracy of the proposed RAPID-LSSVM-I and RAPID-LSSVM-II models compared to baseline methods on the ecoli-0-1-4-6_vs_5, ecoli-0-3-4-7_vs_5-6, votes, and yeast3 datasets. As the proportion of noisy labels increases, the performance of baseline models fluctuates significantly and degrades rapidly. In contrast, the proposed RAPID-LSSVM models maintain stable and consistently superior performance, confirming the effectiveness of the radius-adaptive proximity weighting and probability-aware refinement in suppressing noisy and ambiguous samples. To examine the influence of the regularization parameter C and the kernel width σ, we visualize the classification accuracy using three-dimensional surface plots generated via MATLAB. Figure 6.4 presents the accuracy surfaces for the brwisconsin and breast_cancer_wisc datasets using both RAPID-LSSVM-I and RAPID-LSSVM-II. As observed, the accuracy surfaces exhibit broad plateau regions with high performance over a wide range of C and σ values. This indicates that, although these parameters influence performance, the proposed RAPID-LSSVM models do not exhibit abrupt or erratic degradation with respect to C and σ, reflecting stable behavior similar to classical kernel-based learning methods. Figure 6.5 investigates the impact of the flexibility parameter λ and the k-nearest neighborhood size k on the performance of the proposed RAPID-LSSVM models for the ecoli-0-6-7_vs_3-5 and statlog-german-credit datasets. In contrast to the C–σ surfaces, the accuracy landscapes with respect to (λ, k) exhibit more pronounced variations, with clear regions of performance degradation and improvement. These plots indicate that the classification accuracy is more sensitive to the joint selection of λ and k, and suboptimal choices can lead to noticeable reductions in performance. This behavior is consistent with the functional role of these parameters in the RAPID weighting mechanism. The flexibility parameter λ determines the extent of the plateau region in the radius-adaptive proximity function, thereby controlling how uniformly samples near the class center are weighted before decay begins. Meanwhile, the neighborhood size k gov-

erns the locality of the class-probability estimation $p(x)$, directly affecting how local label consistency is captured. Since both parameters act directly at the sample-weighting level rather than at the global optimization level, their influence on model performance is more pronounced, as reflected in the sharper accuracy variations observed in Figure 6.5.

6.4.7 Practical Scope and Limitations

The RAPID mean- and median-based variants describe each class by one center and one radius. This global radial description can misrepresent multimodal, non-convex, elongated, or heterogeneous classes; the median reduces sensitivity to extremes but not to multimodality. Such settings require multi-prototype, clustering-based, or local-manifold extensions.

The local probability term uses Euclidean k-nearest neighbors. In high dimensions, distances can concentrate and obscure near–far distinctions [134]; reliability can therefore depend on feature scaling, irrelevant dimensions, density, and k. Dimensionality reduction, metric learning, adaptive neighborhoods, or approximate search may be needed.

Tuning (λ, k) is not cost-free. The grid contains up to 10×10 combinations; the brute-force implementation in Section 6.3.3 has time $\mathcal{O}(n^2 d + n^2 \log n)$, memory $\mathcal{O}(n^2)$, and repeated kernel-system costs. Its overhead for large n or d, relative to a deep network, is implementation- and dataset-dependent.

The sensitivity plots provide empirical starting ranges, not universal defaults: $2 \leq \lambda \leq 5$ and $3 \leq k \leq 7$ often perform well here, but the best values remain data-dependent. Together with the single 70:30 split, this motivates repeated nested validation, adaptive parameter selection, and scalable neighborhood computation in future applications.

6.5 Chapter Summary

In this chapter, we introduced RAPID (Radius-Adaptive, Probability and Imbalance Driven), a robust and flexible weighting framework designed to explicitly mitigate label noise, outliers, and class imbalance, which are pervasive challenges in Alzheimer's disease (AD) diagnosis from real-world clinical data. By embedding RAPID into the least-squares SVM framework, we developed two complementary models: RAPID-LSSVM-I, based on a mean-centered representation, and RAPID-LSSVM-II, which employs a median-centered design for enhanced robustness. The proposed RAPID mechanism integrates three key components: (i) a radius-adaptive proximity term that preserves the influence of informative boundary samples while smoothly down-weighting distant outliers; (ii) a k-NN–based probability refinement that suppresses ambiguous or noisy instances through local class consistency; and (iii) an imbalance-ratio correction that compensates for skewed class priors. Importantly, these robustness enhancements are achieved while retaining the computational efficiency of LSSVM, requiring only the solution of linear systems without quadratic programming. The reported experiments on KEEL, UCI, noisy-label, and ADNI tasks show favorable accuracy and F-measure under the common 70:30 protocol, with Friedman and Nemenyi analyses indicating consistent rank behavior. These results remain conditional on the selected split and parameter search. The single-prototype geometry, high-dimensional k-NN reliability, quadratic brute-force neighborhood cost, and dataset-dependent tuning of λ and k delimit the present framework's scope.

Future work can use metaheuristic or self-adaptive procedures to select λ and k during training, and can integrate RAPID with contemporary models while preserving robustness to noise and class imbalance.

Chapter 7

Robust and Inversion-Free Learning in Randomized Neural Networks

This chapter extends robust loss-function design to randomized neural networks, where fixed random representations enable fast learning but standard squared-error objectives can be sensitive to noise, outliers, corrupted labels, and matrix-inversion instability. The first part develops the fleXi Guardian random vector functional link (XG-RVFL) framework. It introduces the fleXi Guardian (XG) loss, a bounded, smooth, dynamically asymmetric loss that adaptively penalizes positive and negative deviations, and embeds it into random vector functional link (RVFL) learning through an inversion-free optimization strategy for both binary and multiclass classification. Theoretical properties of the XG loss and a Rademacher-complexity-based generalization bound for XG-RVFL are also established. The second part develops the Wave loss-based Broad Learning System (Wave-BLS), which replaces the standard least-squares broad learning objective with a bounded and smooth Wave-loss objective solved by a Nesterov accelerated gradient scheme without matrix inversion. Together, these two developments show how robust loss functions can improve randomized neural networks while retaining their computational efficiency. The chapter validates the proposed models on broad benchmark collections and controlled uncertainty settings, with statistical tests

confirming their advantages over existing randomized learning baselines.

7.1 XG-RVFL Framework

7.1.1 Introduction

Artificial neural networks (ANNs) have gained prominence in machine learning due to their remarkable ability to model complex nonlinear relationships, establishing them as a pivotal tool across a wide range of tasks [14], [15]. ANNs have demonstrated their utility in many applications, including video segmentation [135], medical diagnosis [136], [137], human behaviour analysis [138], and brain age estimation [110], etc.

Despite their widespread success, ANNs face several fundamental challenges. A key drawback is their reliance on the backpropagation (BP) algorithm for parameter optimization, which, while effective, suffers from computational inefficiency, sensitivity to initialization, and a tendency to converge to suboptimal local minima [139]. Additionally, ANNs require extensive hyperparameter tuning, which significantly affects performance and stability. These limitations underscore the need for alternative approaches and improved optimization strategies to enhance ANN robustness and efficiency [20].

Randomized neural networks (RaNNs) [140, 141] provide an efficient alternative by fixing hidden-layer parameters and learning only the output weights. Among these models, the random vector functional link (RVFL) network [16, 118] is widely used because it combines random hidden features with direct input-to-output links. In the present chapter, the RVFL structure is used as the host model, and the focus is on three limitations that affect its reliability in real-world data:

1. The RVFL framework relies on the squared error loss function (l_2 loss), which, while computationally convenient, is highly sensitive to noise and outliers.

2. RVFL requires matrix inversion to compute output parameters, leading to significant computational and memory overhead, thereby limiting its scalability for large-scale or high-dimensional data.

3. Existing robust RVFL variants do not simultaneously provide a unified binary/multiclass formulation with flexible asymmetric penalties.

A variety of RVFL variants have been proposed in the literature, each incorporating unique features and enhancements to address the aforementioned limitations. To address the sensitivity of the RVFL framework to noise and outliers, researchers have proposed various variants. These variants can be broadly categorized into two approaches: (1) integrating fuzzy-based frameworks to handle uncertainty and mitigate the effects of noise [101], [55], [56] and (2) replacing the traditional squared error loss function with robust alternatives that are less influenced by extreme values [142], [143]. These advancements aim to improve the RVFL network's resilience in noisy and real-world environments. Beyond addressing noise and outliers, scalability has emerged as another significant challenge for the RVFL framework. To overcome this limitation, Akhtar et al. [143] introduced HE-RVFL, a model that leverages the HawkEye loss function [144] while reformulating the optimization process to entirely eliminate the need for matrix inversion. Despite this advancement, HE-RVFL faces two critical limitations. First, its formulation is designed exclusively for binary classification and lack extensions for multiclass problems, limiting their applicability in broader scenarios. Second, the inherent symmetry of the HawkEye loss function limits its ability to treat positive and negative deviations differently, making it less effective in adapting to dataset-specific characteristics where asymmetric treatment of errors is crucial for improved robustness and generalization.

To address the limitation of symmetry, numerous asymmetric loss functions have been developed in the literature, including the wave loss [67], LINEX loss [31], guardian loss

[107], and others. The guardian loss function is specifically designed to tackle challenges in support vector machines (SVMs) by inherently assigning higher penalties to misclassified samples (right deviations, i.e., samples falling on the wrong side of the optimal hyperplane) while maintaining lower penalties for correctly classified samples (left deviations, i.e., samples correctly positioned relative to the optimal hyperplane). While this fixed asymmetry is well-suited for SVMs, it lacks the flexibility required in scenarios where left deviations hold equal importance as right deviations. This rigidity limits its adaptability to broader machine learning models beyond SVMs.

Building upon this foundation, we introduce the fleXi guardian (XG) loss function (a flexible extension of the guardian loss), designed to enhance both robustness and adaptability. The proposed XG loss is characterized by dynamic asymmetry and boundedness, allowing it to assign different penalties to various types of deviations and adjust penalty magnitudes based on dataset characteristics. By embedding the proposed XG loss into the RVFL framework, we develop XG-RVFL (random vector functional link network based on fleXi guardian loss), a robust and unified model capable of handling both binary and multiclass classification tasks. Crucially, the optimization problem for determining the XG-RVFL model parameters is formulated to eliminate the computational burden of matrix inversion, ensuring scalability for large-scale and high-dimensional datasets.

The key highlights of the work are as follows:

- We propose a novel loss function, fleXi guardian (XG) loss, epitomized by its key properties of dynamic asymmetry and robustness, alongside with smoothness.
- We fuse the XG loss into the RVFL framework and formulate XG-RVFL, a robust and unified model to handle both binary and multiclass classification tasks. The optimization problem of XG-RVFL obviates the necessity for matrix inversion and enhances scalability. Furthermore, we derive a generalization error bound for XG-RVFL using

Rademacher complexity theory, offering formal performance guarantees.

- This chapter stands as the first in the RVFL literature to present a framework that effectively resolves the following limitations of the RVFL simultaneously: (i) sensitivity to noise and outliers, (ii) computational and memory inefficiencies caused by matrix inversion, and (iii) unified framework for both binary and multiclass tasks.
- We establish a rigorous generalization error bound for XG-RVFL using Rademacher complexity, which quantifies expected generalization performance by linking hypothesis class complexity and the Lipschitz continuity of the XG loss.
- We provide comprehensive experimental validation, demonstrating that the proposed XG-RVFL consistently outperforms existing RVFL variants across both binary and multiclass classification tasks. Statistical analyses using the Friedman and Nemenyi post-hoc test confirm the significance of these improvements.

The remainder of the XG-RVFL part first fixes the notation used in this chapter, then introduces the XG loss, the inversion-free XG-RVFL formulation, the generalization analysis, and the experimental validation. The second part of the chapter follows the same pattern for Wave-BLS.

7.1.2 Background and Notation

This section fixes only the notation and the immediate motivation needed for XG-RVFL. Let $X \in \mathbb{R}^{m\times d}$ denote the input matrix, $Y \in \mathbb{R}^{m\times c}$ the target matrix, $H \in \mathbb{R}^{m\times h}$ the randomized hidden-layer output matrix, and $Z = [X, H] \in \mathbb{R}^{m\times(d+h)}$ the augmented RVFL representation. The trainable output weight matrix is denoted by $\Theta \in \mathbb{R}^{(d+h)\times c}$.

Standard RVFL learning estimates Θ through a squared-error least-squares objective on $Z\Theta - Y$. This is computationally efficient, but it has two limitations that are central to

this chapter: squared-error fitting is sensitive to large deviations, and the usual closed-form solution requires matrix inversion. Therefore, the proposed framework keeps the RVFL representation but replaces the squared-error objective with a bounded and flexible loss optimized without direct matrix inversion.

The immediate starting point for the proposed XG loss is the Guardian loss [107], which is bounded and asymmetric:

$$\mathcal{L}_G(\xi) = \frac{\xi\{\exp(a\xi) - 1\}}{1 + \xi\{\exp(a\xi) - 1\}}, \ \forall\, \xi \in \mathbb{R}, \tag{7.1}$$

where $a > 0$ fixes the direction of asymmetry. While this behavior is suitable for SVM-type margin penalties, RVFL output errors may require flexible treatment of positive and negative deviations, as well as tunable control of the upper bound. These two requirements motivate the fleXi Guardian loss introduced next.

7.1.3 Proposed XG-RVFL Framework

In this section, we present a flexible extension of the guardian loss function, termed the fleXi guardian (XG) loss. Building on this foundation, we embed the XG loss function into the RVFL framework and propose a robust and scalable model referred to as XG-RVFL, i.e., the random vector functional link neural network based on the XG loss function. We begin by presenting the mathematical formulation and core properties of the fleXi guardian loss. Subsequently, we detail the formulation of the XG-RVFL model, including its architecture and the optimization algorithm employed for efficient training. In addition, we provide a generalization error bound for XG-RVFL derived using Rademacher complexity theory and analyze the computational complexity of the proposed optimization procedure.

7.1.3.1 FleXi Guardian (XG) Loss

The guardian loss function [107] handles asymmetric penalties by assigning higher penalties to right deviations compared to left deviations, which is well-suited for SVMs where right-side errors correspond to misclassified samples, and left-side deviations indicate correctly classified ones. However, this fixed asymmetry limits its adaptability for models such as RVFL, where deviations on both sides are important. Additionally, the guardian loss has a rigid bound of 1, restricting its applicability in scenarios requiring dynamic control over penalty magnitudes. The limitations of the guardian loss function necessitate a more flexible and adaptable approach to handle diverse scenarios where both-sided deviations and dynamic bounds are critical. To address these challenges, we propose the fleXi guardian (XG) loss, which extends the guardian loss by introducing two key features:

- **Dynamic asymmetry:** The XG loss allows assigning higher penalties to deviations on either side, enabling it to adapt to tasks where left-sided deviations are as important as right-sided ones.
- **Dynamic bound control:** A parameter $\eta > 0$ is incorporated to adjust the upper bound of the loss, providing the flexibility to scale penalty magnitudes as required.

The XG loss function is mathematically defined as:

$$\mathcal{L}_{XG}(\xi) = \frac{a\xi\{\exp(a\xi) - 1\}}{1 + a\eta\xi\{\exp(a\xi) - 1\}}, \ \forall\, \xi \in \mathbb{R}, \tag{7.2}$$

where ξ represents the deviation, $a \neq 0$ governs both the steepness and the asymmetry of the loss function, and $\eta > 0$ adjusts the upper bound. Figure 7.1 depicts the visual illustration of the proposed XG loss function under varying parameter configurations. Figures 7.1a, 7.1b, 7.1c, and 7.1d separately show the effects of positive and negative a values and the bound-control parameter η.

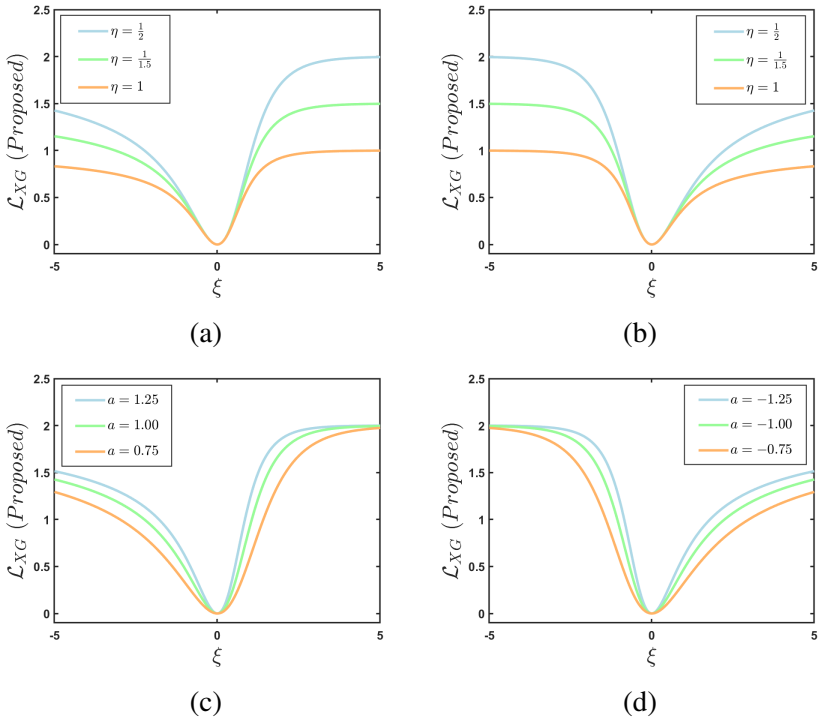


Figure 7.1: Illustrate the behavior of the proposed XG loss function under varying parameter configurations. Subfigures (a) and (b) are plotted for fixed values of a ($a = 1$ in (a) and $a = -1$ in (b)) while varying the bound-control parameter η. Subfigures (c) and (d) are plotted for a fixed $\eta = 1/2$ while varying a. Subfigure (c) explores positive values of a, showcasing its effect on the steepness and penalties for positive deviations, whereas subfigure (d) examines negative values of a, highlighting its influence on negative deviations.

7.1.3.2 Theoretical Properties of the FleXi Guardian (XG) Loss

We now present the key theoretical properties of the proposed XG loss function. These include asymmetry, boundedness, smoothness, Lipschitz continuity, and robustness, all of which contribute to its adaptability and stability across diverse learning settings.

Asymmetry The proposed XG loss introduces dynamic asymmetry, allowing the penalty distribution to adjust flexibly based on the parameter $a \neq 0$. This parameter plays a pivotal role in determining the emphasis on penalties for deviations on either side. Specifically, for $a > 0$, the loss grows more rapidly for positive residuals, placing greater penalty on overestimation. Conversely, for $a < 0$, the opposite holds, making it suitable for cases where underestimation is more critical. This asymmetric behavior enables the loss to adapt to problem-specific risks, such as cost-sensitive misclassification or skewed noise distributions.

Boundedness The proposed XG loss function retains the bounded nature of the original guardian loss while introducing enhanced flexibility through the parameter $\eta > 0$. This boundedness ensures that the loss values remain constrained within a specific threshold, making the model robust to outliers by preventing excessively large penalties from disproportionately influencing the optimization process. The upper bound of the XG loss is dynamically controlled by the parameter η, allowing the maximum penalty to be adjusted based on the dataset's characteristics. Mathematically, the XG loss satisfies the condition:

$$0 \leq \mathcal{L}_{XG}(\xi) \leq \frac{1}{\eta}, \forall\, \xi \in \mathbb{R}. \tag{7.3}$$

Here, $\eta > 0$ determines the maximum threshold of the loss, ensuring penalties remain well-regulated. This property is particularly beneficial in scenarios with noisy or outlier-prone

datasets and contributes to reliable and efficient learning.

Smoothness The proposed XG loss is infinitely differentiable (i.e., smooth), ensuring stability and efficiency in gradient-based optimization. The smoothness arises from the exponential terms in its formulation, allowing gradual changes in the loss values as deviations increase, thus avoiding abrupt shifts or discontinuities. The first derivative of the XG loss with respect to ξ is given by:

$$\frac{d\mathcal{L}_{XG}(\xi)}{d\xi} = \frac{a\big[(a\xi+1)\cdot\exp(a\xi)-1\big]}{\big[1+a\eta\xi\{\exp(a\xi)-1\}\big]^2}. \tag{7.4}$$

This derivative is continuous and bounded on any compact interval. Moreover, it satisfies $\lim_{\xi\to\pm\infty} d\mathcal{L}_{\text{XG}}/d\xi = 0$, ensuring smooth gradient decay at extremes, which contributes to training stability.

Lipschitz Continuity The proposed XG loss is Lipschitz continuous on any compact domain, making it suitable for generalization analysis.

Theorem 7.1.1 (Lipschitz Continuity)**.** *Let $M > 0$. Then $\mathcal{L}_{XG}(\xi)$ is Lipschitz continuous on $[-M, M]$; that is, there exists a constant $L_M > 0$ such that:*

$$|\mathcal{L}_{XG}(\xi_1) - \mathcal{L}_{XG}(\xi_2)| \le L_M|\xi_1 - \xi_2|, \quad \forall \xi_1, \xi_2 \in [-M, M].$$

Proof. The derivative in equation (7.4) is continuous on $[-M, M]$, and by the extreme value theorem, attains a maximum L_M. Then by the mean value theorem, for any $\xi_1, \xi_2 \in [-M, M]$, there exists $\xi^* \in (\xi_1, \xi_2)$ such that

$$|\mathcal{L}_{\text{XG}}(\xi_1) - \mathcal{L}_{\text{XG}}(\xi_2)| = \left|\frac{d\mathcal{L}_{\text{XG}}}{d\xi}(\xi^*)\right| \cdot |\xi_1 - \xi_2| \le L_M|\xi_1 - \xi_2|.$$

Hence, Lipschitz continuity holds on compact intervals. □

Robustness via Influence Function The *influence function* (IF) is a fundamental concept in robust statistics that measures the sensitivity of an estimator to infinitesimal contamination in the data [145], [146]. For loss-based estimators, the IF is often proportional to the derivative of the loss with respect to the residual. A bounded or vanishing IF implies that extreme outliers have limited effect on the estimator, thereby enhancing robustness [147]. For the XG loss, the first derivative is given by:

$$\frac{d\mathcal{L}_{\text{XG}}}{d\xi} = \frac{a\left[(a\xi+1)e^{a\xi}-1\right]}{\left[1+a\eta\xi(e^{a\xi}-1)\right]^2}.$$

It can be established that $\lim_{|\xi|\to\infty} d\mathcal{L}_{\text{XG}}/d\xi = 0$, which implies that the XG loss exhibits *vanishing influence* for large residuals. This behavior ensures that extreme data points exert negligible effect on the learned parameters, promoting robustness in noisy or adversarial settings.

In contrast, traditional losses such as the squared error loss have unbounded influence; for example, $\mathcal{L}(\xi) = \xi^2$ yields $\text{IF}(\xi) = 2\xi$, which grows without bound as $|\xi| \to \infty$. Thus, unlike squared error, the XG loss prevents gradient explosion and resists distortion from heavy-tailed or contaminated data, making it well-suited for reliable learning in real-world applications.

7.1.3.3 Formulation and Optimization of XG-RVFL

In this subsection, we incorporate the proposed XG loss function into the RVFL framework by replacing the conventional squared error loss and develop a robust and scalable framework, termed XG-RVFL. The integration of the XG loss function introduces dynamic asymmetry and boundedness, enabling the framework to effectively handle chal-

lenges posed by outliers and noise, thereby significantly enhancing robustness. To solve the XG-RVFL model, we formulate an optimization problem that avoids the computational overhead of matrix inversion during parameter estimation. This approach ensures scalability, making the XG-RVFL framework suitable for large-scale datasets.

The optimization problem of the proposed XG-RVFL is formulated as follows:

$$\underset{\Theta}{\operatorname{argmin}} \; J(\Theta) = \frac{1}{2}\|\Theta\|^2 + \lambda \cdot \mathcal{L}_{XG}(Z\Theta - Y), \tag{7.5}$$

where $\lambda > 0$ is the regularization parameter and $L_{XG}(Z\Theta - Y)$ represents the total loss, derived from the XG loss function.

The XG loss for a single sample i and class j is computed by:

$$\mathcal{L}_{XG}(\xi_{ij}) = \frac{a\xi_{ij}\{\exp(a\xi_{ij}) - 1\}}{1 + ab\xi_{ij}\{\exp(a\xi_{ij}) - 1\}}, \tag{7.6}$$

where $\xi_{ij} = (Z\Theta)_{ij} - Y_{ij}$ is the error for sample i and class j. The total loss $\mathcal{L}_{XG}(Z\Theta - Y)$ is computed over all samples and classes as:

$$\mathcal{L}_{XG}(Z\Theta - Y) = \sum_{i=1}^{m}\sum_{j=1}^{c}\mathcal{L}_{XG}(\xi_{ij}).$$

Hence, the objective function (7.5) can be rewritten as follows:

$$\underset{\Theta}{\operatorname{argmin}} \; J(\Theta) = \frac{1}{2}\|\Theta\|^2 + \lambda \cdot \sum_{i=1}^{m}\sum_{j=1}^{c}\mathcal{L}_{XG}(\xi_{ij}). \tag{7.7}$$

To optimize the proposed XG-RVFL model, we leverage the smooth characteristic of the objective function (7.7) and adopt the Nesterov Accelerated Gradient (NAG)-based algorithm as used in [90]. Using NAG-based algorithm, the parameter estimation process is designed to avoid the computationally intensive matrix inversion typically required in tradi-

tional RVFL, thereby enhancing computational efficiency and scalability. The adoption of NAG-based algorithm for optimizing the XG-RVFL model is both strategic and impactful, as it not only enhances convergence speed but also ensures stability and robustness during the optimization process, making it well-suited for the proposed framework.

To solve the optimization problem (7.7) using NAG-based algorithm, we derive the gradient of the objective function to facilitate efficient optimization. For a given class $j \in \{1, 2, \ldots, c\}$, the gradient of the XG loss function with respect to the parameter Θ is formulated as follows:

$$\text{grad}_{XG}(:, j) = \sum_{i=1}^{m} \frac{a\big[(a\xi_{ij} + 1) \cdot \exp(a\xi_{ij}) - 1\big]}{\big[1 + a\eta\xi_{ij} \{\exp(a\xi_{ij}) - 1\}\big]^2} \cdot Z(i, :)^\top. \tag{7.8}$$

Note: Algorithm 7.1 provides a detailed outline of the process for calculating the gradient of the XG loss. This step-by-step procedure ensures precise gradient computation by iterating over all samples and classes. The computed gradient plays a pivotal role in the optimization process, enabling efficient parameter updates.

The overall gradient of the objective function $J(\Theta)$ is given by:

$$\nabla J(\Theta) = \Theta + \lambda \cdot \text{grad}_{XG}. \tag{7.9}$$

Here, the first term, Θ, arises from the regularization term $\frac{1}{2}\|\Theta\|^2$ and the second term, grad_{XG}, reflects the contribution of the XG loss.

At t^{th} iteration, a mini-batch of k samples is randomly selected and utilized to compute the gradient, ensuring efficient and stochastic updates during the optimization process. Now, the step-by-step procedure of NAG-based algorithm to solve the XG-RVFL model (7.7) is as follows:

1. **Initialization:** Initialize the model parameter $\Theta^{(0)}$, velocity vector $v^{(0)}$, learning rate

Algorithm 7.1 Algorithm for computing the gradient of the loss term.

Input: $\beta = Z\Theta - Y \in \mathbb{R}^{m\times c}$: matrix of errors, $Z \in \mathbb{R}^{m\times(d+h)}$: transformed feature matrix.
Output: $\text{grad}_{XG} \in \mathbb{R}^{(d+h)\times c}$: gradient of the XG loss term.

Initialize: $\text{grad}_{XG} = \mathbf{0} \in \mathbb{R}^{(d+)h\times c}$.
for $j \leftarrow 1$ to c **do**
  for $i \leftarrow 1$ to m **do**
    Compute $\text{grad}_{XG}(i, j)$ using equation (7.8).
    Update $\text{grad}_{XG}(:, j) = \text{grad}_{XG}(:, j) + \text{grad}_{XG}(i, j) \cdot Z(i, :)^\top$.
  end for
end for
return grad_{XG}.

$\mu^{(0)}$. Set the momentum coefficient γ, learning rate decay parameter α, and the maximum number of iterations $I_{\max}$.

2. **Iteration:** For each iteration $t = 1, 2, \ldots, I_{\max}$;

   (a) Compute look-ahead point:

$$\Theta^{\text{look-ahead}} = \Theta^{(t)} + \gamma \cdot v^{(t)}.$$

   (b) Calculate the gradient at look-ahead point using equation (7.9):

$$g^{(t)} = \nabla J(\Theta^{\text{look-ahead}}).$$

   (c) Update velocity vector:

$$v^{(t+1)} = \gamma \cdot v^{(t)} - \mu^{(t)} \cdot g^{(t)}.$$

   (d) Update XG-RVFL parameter:

$$\Theta^{(t+1)} = \Theta^{(t)} + v^{(t+1)}.$$

(e) Adjust learning rate with exponential decay:

$$\mu^{(t+1)} = \mu^{(t)} \cdot \exp(-\alpha t).$$

3. **Stopping Criteria:** The iteration process is terminated if either of the following criteria is met: (a) the maximum number of iterations is reached, or (b) the change in Θ falls below a predefined threshold, indicating convergence.

Note: The detailed process of calculating Θ for the proposed XG-RVFL model is meticulously outlined in algorithm (7.2).

Algorithm 7.2 NAG-based optimization for XG-RVFL

Input: $Z \in \mathbb{R}^{m\times(d+h)}$: transformed feature matrix, $Y \in \mathbb{R}^{m\times c}$: target matrix, μ: learning rate, α: learning rate decay factor, γ: momentum parameter, λ: regularization parameter, $I_{\max}$: maximum number of iterations, δ: convergence tolerance.

Output: $\Theta \in \mathbb{R}^{(d+h)\times c}$: Output layer parameters.

1: **Initialize:** Set iteration $t = 0$, initialize $\Theta^{(0)}$, $v^{(0)}$, and $\mu^{(0)}$.

2: **repeat**

3: **Look-ahead step:**

$$\Theta^{\text{look-ahead}} = \Theta^{(t)} + \gamma \cdot v^{(t)}.$$

4: **Compute the gradient:** For mini-batch of size k calculate the gradient $g^{(t)}$ at look-ahead point using (7.9).

5: **Update the velocity:**

$$v^{(t+1)} = \gamma \cdot v^{(t)} - \mu^{(t)} \cdot g^{(t)}.$$

6: **Update the model parameters:**

$$\Theta^{(t+1)} = \Theta^{(t)} + v^{(t+1)}.$$

7: **Update learning rate :**

$$\mu^{(t+1)} = \mu^{(t)} \cdot \exp(-\alpha t).$$

8: $t \leftarrow t + 1$.

9: **until** Convergence criteria are met: $t \geq I_{\max}$ or $\|\Theta^{(t+1)} - \Theta^{(t)}\| \leq \delta$.

10: **Return** Θ.

After obtaining the optimal Θ, the following decision function is employed to predict

the label vector of the test set $(\tilde{X})$:

$$\tilde{y} = \text{sign}\left(\tilde{Z}\Theta\right),$$

where $\tilde{Z}$ is the concatenation matrix comprising the test data and output from the hidden layer.

7.1.3.4 Generalization Bound for the Proposed XG-RVFL

We now derive a generalization bound for the proposed XG-RVFL model using Rademacher complexity. The bound quantifies the expected performance of the model on unseen data, in terms of the network complexity and the properties of the XG loss.

Let the training data be given by $\{(x_i, y_i)\}_{i=1}^m$, where each input $x_i \in \mathbb{R}^d$ and each target $y_i \in \mathbb{R}^c$ is a one-hot encoded vector representing one of c classes. Let $Z(x) \in \mathbb{R}^{d+h}$ denote the fixed feature mapping applied in RVFL, defined by $Z(x) = [x^\top, h(x)^\top]^\top$, where $h(x)$ is the hidden layer output after nonlinear transformation and h is the number of hidden nodes. The model predicts a vector-valued output:

$$f_\Theta(x) = Z(x)^\top \Theta \in \mathbb{R}^c, \quad \text{where } \Theta \in \mathbb{R}^{(d+h)\times c}.$$

Let the hypothesis class be defined as:

$$\mathcal{F}_B = \left\{ f_\Theta(x) = Z(x)^\top \Theta \ : \ \|\Theta\|_F \leq B \right\},$$

where $\|\Theta\|_F$ denotes the Frobenius norm, and we assume $\|Z(x)\|_2 \leq R$ for all x. The total loss is defined element-wise over each output coordinate using the scalar XG loss:

$$\mathcal{L}_{XG}(f(x), y) = \sum_{j=1}^{c} \mathcal{L}_{XG}(f_j(x) - y_j),$$

where $f_j(x)$ is the jth coordinate of $f(x)$, and $\mathcal{L}_{XG}(\cdot)$ is the scalar loss from equation (7.2), which is assumed to be L-Lipschitz and bounded by $M > 0$.

Theorem 7.1.2 (Generalization Bound for XG-RVFL)**.** *Let the XG loss be applied element-wise as described above. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following bound holds for all $f \in \mathcal{F}_B$:*

$$\mathbb{E}_{(x,y)}[\mathcal{L}_{XG}(f(x), y)] \leq \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{XG}(f(x_i), y_i) + \frac{2LBRc}{\sqrt{m}} + 3Mc\sqrt{\frac{\log(1/\delta)}{2m}}.$$

Proof. Define the composed function class:

$$\mathcal{L}_{XG} \circ \mathcal{F}_B = \left\{ (x, y) \mapsto \sum_{j=1}^{c} \mathcal{L}_{XG}(f_j(x) - y_j) \ : \ f \in \mathcal{F}_B \right\}.$$

By the additivity of the scalar XG loss over output dimensions and the application of the vector-valued contraction inequality (see [87]), we obtain:

$$\mathfrak{R}_m(\mathcal{L}_{XG} \circ \mathcal{F}_B) \leq \sum_{j=1}^{c} L \cdot \mathfrak{R}_m(\mathcal{F}_B^{(j)}) = L \cdot \sum_{j=1}^{c} \mathfrak{R}_m(\mathcal{F}_B^{(j)}),$$

where $\mathcal{F}_B^{(j)}$ is the class of scalar functions $f_j(x) = Z(x)^\top \Theta_j$, corresponding to the jth column of Θ. Since each Θ_j satisfies $\|\Theta_j\|_2 \leq B$ (implied by $\|\Theta\|_F \leq B$), we have:

$$\mathfrak{R}_m(\mathcal{F}_B^{(j)}) \leq \frac{BR}{\sqrt{m}}.$$

Hence:

$$\mathfrak{R}_m(\mathcal{L}_{XG} \circ \mathcal{F}_B) \leq \frac{LBRc}{\sqrt{m}}.$$

Applying the standard Rademacher complexity generalization bound for bounded losses

(see [148]), we conclude that with probability at least $1-\delta$, for all $f \in \mathcal{F}_B$:

$$\mathbb{E}[\mathcal{L}_{XG}(f(x), y)] \leq \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_{XG}(f(x_i), y_i) + \frac{2LBRc}{\sqrt{m}} + 3Mc\sqrt{\frac{\log(1/\delta)}{2m}}.$$

This completes the proof. □

7.1.3.5 Computational Complexity

The computational complexity of the proposed XG-RVFL model is governed by the use of the NAG-based optimization algorithm. During each iteration t, the gradient of the objective function $\nabla J(\Theta)$ is calculated. This involves evaluating the gradient term defined in equation (7.9) by processing a mini-batch of k samples across c output classes. For each sample i, the computation requires a dot product between the transformed feature matrix $Z(i,:)$, with dimensions $(d+h)$, and its corresponding scalar term. As a result, the gradient computation incurs a cost of $\mathcal{O}(k \cdot c \cdot (d+h))$. After the gradient is obtained, the NAG framework performs three main operations: a look-ahead step, a velocity update, and a parameter update. Each of these steps involves matrix computations with dimensions $(d+h)\times c$, contributing an additional cost of $\mathcal{O}((d+h)\cdot c)$ per iteration. However, since gradient computation dominates the overall cost, the total computational complexity for a single iteration is approximately $\mathcal{O}(k \cdot c \cdot (d+h))$. For $I_{\max}$ iterations, the total complexity of the proposed XG-RVFL becomes $\mathcal{O}(I_{\max}\cdot k\cdot c\cdot(d+h))$. Unlike traditional RVFL models, which involve a computationally expensive matrix inversion step with a complexity of $\mathcal{O}(\min((d+h)^3, m^3))$, the proposed approach eliminates this requirement. By avoiding matrix inversion, the proposed XG-RVFL model achieves improved scalability and computational efficiency, making it well-suited for large-scale datasets.

Avoiding explicit inversion removes a cubic algebraic bottleneck, but it does not guarantee lower runtime for every dataset. The practical advantage depends on $I_{\max}$, mini-batch

size k, learning-rate and momentum choices, stopping tolerance, and the conditioning of the iterative optimization problem. XG-RVFL is computationally favorable when the work needed to reach an acceptable solution, approximately $\mathcal{O}(I_{\max}kc(d+h))$, remains below the cost and memory burden of the corresponding factorization or inversion. Poor conditioning or a large iteration count can narrow or reverse this advantage. Thus, inversion-free denotes the absence of an explicit inverse, while scalability is conditional on convergence behavior.

7.1.4 Experiments, Results and Discussion

In this section, we first present details of the datasets, compared models, and experimental setup. Furthermore, to evaluate the performance of the proposed XG-RVFL, we analyze the results and conduct statistical assessments for rigorous validation.

Dataset Description and Compared Models

To evaluate the performance of the proposed XG-RVFL model, we employ 86 benchmark datasets from the UCI [80] and KEEL [79] repositories, spanning diverse domains and sample sizes. These datasets are categorized into 37 binary UCI classification tasks, 25 multiclass UCI classification tasks, and 24 binary KEEL datasets. For binary UCI classification, the number of samples ranges from 100 to 130,064, with feature dimensions varying between 3 and 166. To facilitate nuanced analysis, UCI binary datasets are further divided into two subgroups: B1, containing datasets with fewer than 1,000 samples, and B2, comprising those with 1,000 or more samples. Similarly, multiclass UCI classification datasets span from 103 to 12,960 samples and include 3 to 100 features. These are also split into M1 (samples $<$ 1,000) and M2 (samples $\geq$ 1,000). The number of target classes in multiclass datasets ranges from 3 to 18, ensuring a diverse and comprehensive evaluation of model performance across varying data distributions. Table 7.1 provides details of each binary and multiclass UCI dataset.

We compare the proposed XG-RVFL model against several benchmark methods to assess its effectiveness. These include RVFL [16], RVFLwoDL (RVFL without direct link, also known as extreme learning machine) [17], Total-Var-RVFL (total variance minimization-based RVFL) [149], IF-RVFL (intuitionistic fuzzy RVFL) [101], NF-RVFL-K, NF-RVFL-R, and NF-RVFL-C (neuro-fuzzy RVFL, where fuzzy layer centers are determined using K-means clustering, R-means clustering, and fuzzy C-means clustering, respectively) [55], and HE-RVFL (RVFL based on HawkEye loss) [143].

Experimental Setup

All experiments are implemented in MATLAB R2023a and executed on a Windows 10 PC equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 4 cores (8 logical processors), and 16 GB of RAM. Each dataset is preprocessed by normalizing the input features to have zero mean and unit variance. A 5-fold cross-validation procedure is employed to provide a consistent cross-validated comparison. In each fold, the dataset is split into 80% training data and 20% testing data. For every combination of hyperparameters, the model is trained on the training data and evaluated on the testing data across all 5 folds. The testing accuracy is recorded for each fold. The final testing accuracy for each dataset is computed as the mean testing accuracy across the five folds, providing a robust estimate of the model's performance. The hyperparameter configurations for all baseline models are set identically as set in [55]. For all the models, we use a consistent set of six activation functions: Sigmoid, Sine, Tansig, Tribas, Radbas, and ReLU. For the proposed XG-RVFL, the parameters a and η are tuned within the ranges $[-2 : 0.5 : 2] \setminus \{0\}$ and $[0.5 : 0.5 : 2]$, respectively. The NAG-based algorithm parameters are experimentally set as follows: (i) initial model parameter $\Theta^{(0)} = 0.01 \times \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{(d+h)\times c}$ is a matrix of ones, (ii) initial velocity $v^{(0)} = \mathbf{0}$, where $\mathbf{0} \in \mathbb{R}^{(d+h)\times c}$ is a matrix of zeros, (iii) initial learning rate $\mu^{(0)} = 0.01$, (iv) learning rate decay factor $\alpha = 0.1$, (v) momentum coefficient $\gamma = 0.6$, (vi) mini-batch size

$k = 2^6$, and (vii) maximum number of iterations $I_{\max} = 500$.

Table 7.1: Details of binary and multiclass datasets, including the number of samples, features, and target classes.

Dataset	# of Samples	# of Features	# of Classes
Binary (B1)			
acute_inflammation	120	6	2
acute_nephritis	120	6	2
blood	748	4	2
breast_cancer	286	9	2
breast_cancer_wisc_prog	198	33	2
congressional_voting	435	16	2
conn_bench_sonar_mines_rocks	208	60	2
credit_approval	690	15	2
cylinder_bands	512	35	2
echocardiogram	131	10	2
fertility	100	9	2
haberman_survival	306	3	2
heart_hungarian	294	12	2
hepatitis	155	19	2
ilpd_indian_liver	583	9	2
molec_biol_promoter	106	57	2
musk_1	476	166	2
parkinsons	195	22	2
pima	768	8	2
pittsburg_bridges_T_OR_D	102	7	2
planning	182	12	2
spect	265	22	2
spectf	267	44	2
statlog_heart	270	13	2
tic_tac_toe	958	9	2
vertebral_column_2clases	310	6	2
Binary (B2)			
chess_krvkp	3196	36	2
connect_4	67557	42	2
magic	19020	10	2
miniboone	130064	50	2
musk_2	6598	166	2
ozone	2536	72	2
ringnorm	7400	20	2
spambase	4601	57	2
statlog_german_credit	1000	24	2
titanic	2201	3	2
twonorm	7400	20	2

Dataset	# of Samples	# of Features	# of Classes
Multiclass (M1)			
audiology_std	196	59	18
dermatology	366	34	6
ecoli	336	7	8
glass	214	9	6
hayes_roth	160	3	3
heart_cleveland	303	13	5
heart_switzerland	123	12	5
heart_va	200	12	5
iris	150	4	3
led_display	1000	7	10
low_res_spect	531	100	9
lymphography	148	18	4
pittsburg_bridges_MATERIAL	106	7	3
pittsburg_bridges_REL_L	103	7	3
pittsburg_bridges_TYPE	105	7	6
seeds	210	7	3
synthetic_control	600	60	6
vertebral_column_3clases	310	6	3
Multiclass (M2)			
car	1728	6	4
contrac	1473	9	3
image_segmentation	2310	18	7
molec_biol_splice	3190	60	3
nursery	12960	8	5
wine_quality_red	1599	11	6
wine_quality_white	4898	11	7

7.1.4.1 Evaluation on UCI Binary Datasets

Table 7.2 presents the classification results of the proposed XG-RVFL model compared to baseline methods across 37 binary classification datasets. The results are further organized into two subgroups: B1, containing datasets with fewer than 1,000 samples, and B2, comprising datasets with 1,000 or more samples, enabling a detailed comparison across varying data scales. For the B1 category, the proposed XG-RVFL model achieves the highest average accuracy of 84.6633%, outperforming all baseline models. The closest competitor, HE-RVFL, achieves an accuracy of 82.8046%, followed by NF-RVFL-C, NF-RVFL-K, and Total-var-RVFL with accuracies of 79.4587%, 79.3354%, and 79.2033%, respectively. Traditional RVFL and RVFLwoDL exhibit lower accuracies of 78.0448% and

Table 7.2: Accuracy of the proposed XG-RVFL model compared to baseline models on each of the binary datasets in categories B1 and B2.

Dataset ↓ | Model →	**RVFL** [16]	**RVFLwoDL** [17]	**Total-var-RVFL** [149]	**IF-RVFL** [101]	**NF-RVFL-K** [55]	**NF-RVFL-R** [55]	**NF-RVFL-C** [55]	**HE-RVFL** [143]	**XG-RVFL**†
B1-Category									
acute_inflammation	100	100	100	100	100	100	100	100	100
acute_nephritis	100	100	100	100	100	100	100	92.5	100
blood	76.5065	76.9056	76.9065	77.4398	77.8398	77.838	77.4398	77.6738	77.9732
breast_cancer_wisc_prog	81.359	80.3846	82.359	78.359	81.8846	83.359	80.9103	76.3163	80.3846
breast_cancer	70.1754	70.1754	70.1754	71.9298	72.6921	72.3351	70.8953	95.4861	90.1754
congressional_voting	63.6782	63.2184	63.908	58.8506	63.908	63.908	64.1379	61.3766	64.1379
conn_bench_sonar_mines_rocks	62.079	60.5226	64.4251	54.8316	66.2602	65.8885	63.4611	96.6346	93.6585
credit_approval	85.2174	85.3623	85.5072	86.5217	85.942	85.942	85.7971	96.9292	86.9565
cylinder_bands	66.4154	65.8081	67.7727	63.4875	69.724	68.9606	70.1294	66.7969	68.3229
echocardiogram	84.6724	83.9031	85.4701	80.7692	85.4416	85.4416	85.4416	72.6089	86.9801
fertility	91	91	92	92	92	91	91	90	91
haberman_survival	73.4902	73.8181	74.4738	75.1348	75.1296	75.1296	75.4574	74.4959	74.8017
heart_hungarian	73.8223	74.4886	75.8738	76.8849	76.5576	76.9024	78.2466	91.8919	96.2712
hepatitis	85.1613	82.5806	85.8065	85.8065	87.0968	85.8065	85.8065	82.4561	86.4516
ilpd_indian_liver	71.5311	72.2149	72.5626	72.7277	72.3917	72.3932	72.2178	71.5305	72.7336
molec_biol_promoter	72.7706	74.632	78.2684	78.3983	82.0779	82.0346	82.0346	100	96.1905
musk_1	72.0614	69.7675	72.4846	71.864	76.0482	75.2149	74.1645	93.4874	96.6316
parkinsons	80.5128	80.5128	82.0513	78.4615	83.0769	83.0769	84.1026	79.4537	85.1282
pima	72.0075	72.2698	72.92	73.8282	72.9174	73.6983	72.6594	65.3646	70.3115
pittsburg_bridges_T_OR_D	87.1905	87.1905	89.1429	89.1905	88.2381	90.1905	90.1905	90.1538	90.2857
planning	71.3814	71.3814	72.4925	69.8048	71.9369	73.018	71.9369	73.5749	71.9369
spect	68.3019	66.7925	69.0566	68.3019	69.434	69.0566	68.6792	64.5353	66.7925
spectf	79.3431	79.7205	79.7135	79.3431	79.7205	79.7205	79.7205	81.6316	81.2369
statlog_heart	80.3704	80	81.4815	81.8519	82.2222	81.4815	81.8519	74.4403	81.1111
tic_tac_toe	88.8264	88.9278	91.8521	81.4316	68.5684	65.3125	79.3216	90.3125	95
vertebral_column_2clases	71.2903	70.6452	72.5806	85.4839	81.6129	79.0323	80.3226	93.2692	96.7742
Average Accuracy	78.0448	77.7778	79.2033	78.1809	79.3354	79.1054	79.4587	<u>82.8046</u>	**84.6633**
B2-Category									
chess_krvkp	72.0313	71.936	73.7833	72.6262	81.7908	82.6045	85.2322	89.6508	82.5039
connect_4	75.4518	75.4059	75.4992	75.3407	75.5022	75.4844	76.2156	81.858	75.4859
magic	78.7487	78.5279	78.7171	77.5289	76.3302	76.3407	78.4017	89.858	95.163
miniboone	81.4984	79.2741	81.8659	83.3469	84.0217	84.3969	89.8363	94.9292	93.2441
musk_2	84.5909	84.5909	85.1818	84.5909	83.4091	83.9394	85.9545	90.4091	95.4091
ozone	97.1217	97.1217	97.1611	97.1612	97.1612	97.2005	97.1612	96.806	97.1612
ringnorm	51.5541	51.5405	52.0405	51.473	51.6486	51.6081	51.6892	51.9865	51.8378
spambase	88.546	87.0472	88.6116	85.0883	88.9583	88.9582	89.3931	90.7609	99.3913
statlog_german_credit	76.9	75.7	77	76.7	77.7	77.1	77.7	70.5	73.3
titanic	77.9168	77.9168	78.6901	79.0532	79.0532	78.4623	79.0532	80.5971	79.0532
twonorm	51.5	51.5676	51.6892	51.2027	51.3649	51.7973	51.6622	51.7027	51.8378
Average Accuracy	75.9872	75.5117	76.3854	75.8284	76.9946	77.0811	78.3908	<u>80.8235</u>	**81.3079**
Overall Accuracy	77.4331	77.1041	78.3655	77.4815	78.6395	78.5036	79.1412	<u>82.2156</u>	**83.6658**
Overall Rank	6.97	7.23	4.81	5.95	4.14	4.24	<u>4.01</u>	4.59	**3.05**

Here, bold indicates the best, underlined marks the second-best, and † denotes the proposed model.

77.7778%, respectively, highlighting the improvements introduced by the XG loss. In the B2 category, which consists of larger datasets, the proposed XG-RVFL again outperforms all baseline models with an average accuracy of 81.3079%. The second-best performance is shown by HE-RVFL at 80.8235%, while NF-RVFL-C and NF-RVFL-R follow with accuracies of 78.3908% and 77.0811%, respectively. Other baselines such as RVFL, Total-Var-RVFL, and RVFLwoDL achieve relatively lower accuracies of 75.9872%, 76.3854%, and 75.5117%, respectively. Considering the overall binary classification performance across both B1 and B2 datasets, the proposed XG-RVFL achieves the highest average accuracy of 83.6658 which significantly surpasses the second-best model, HE-RVFL, which obtains 82.2156%. The performance of other methods ranges from 77.1041% (RVFLwoDL) to 79.1412% (NF-RVFL-C), illustrating the consistent superiority of XG-RVFL. Beyond accuracy, the overall average rank provides additional insight into model consistency across datasets. The proposed XG-RVFL achieves the best average rank of 3.05, confirming its stability and superior generalization across diverse datasets. NF-RVFL-C attains the second-best ranking of 4.01, followed by NF-RVFL-K and NF-RVFL-R, which rank 4.14 and 4.24, respectively. Total-Var-RVFL, despite its competitive accuracy, holds a slightly lower ranking of 4.81, while HE-RVFL, with a high accuracy of 82.75%, ranks 4.59, indicating slight variations in its dataset-wise performance. IF-RVFL and RVFL exhibit moderate performance with ranks of 5.95 and 6.97, respectively. RVFLwoDL ranks the lowest at 7.23, further demonstrating its weaker performance compared to RVFL models with direct connections. These results clearly demonstrate that the proposed XG-RVFL model not only achieves the highest average accuracy in both B1 and B2 dataset categories but also maintains the most consistent performance across diverse binary datasets. This superiority is attributed to the dynamic asymmetry and boundedness introduced by the XG loss function, which enhance the model's robustness to noise and outliers while eliminating the computa-

tional burden of matrix inversion.

To further validate the superior performance of the proposed XG-RVFL model against baseline methods, we conduct statistical analyses using the Friedman test and the Nemenyi post-hoc test [25]. The Friedman test, a non-parametric statistical test, is employed to determine whether there are significant differences in performance among the compared models. According to the null hypothesis, all models are considered equivalent, implying that their average ranks do not significantly differ. The Friedman test statistic, which adheres to a chi-squared (χ_F^2) distribution with $l-1$ degrees of freedom (where l represents the total number of models), is computed as $\chi_F^2 = \frac{12\mathcal{D}}{l(l+1)}\left[\sum_i R_i^2 - \frac{l(l+1)^2}{4}\right]$, where $\mathcal{D}$ represents the number of datasets and R_i denotes the mean rank of the ith model. Since the Friedman statistic is often overly conservative, an improved statistic proposed by [62], defined as $F_F = \frac{(\mathcal{D}-1)\chi_F^2}{\mathcal{D}(l-1)-\chi_F^2}$, provides a more accurate measure and follows the F-distribution with degrees of freedom $(l-1)$ and $(l-1)(\mathcal{D}-1)$. At 5% level of significance, the critical value of $F((l-1),(l-1)(\mathcal{D}-1))$ is obtained from the F-distribution table. If $F_F > F((l-1),(l-1)(\mathcal{D}-1))$, the null hypothesis is rejected, indicating significant differences among the models. Table 7.3 reports the results of the Friedman test. For binary datasets, F_F statistic is computed as 13.05, with a corresponding critical value of 1.97. Since the computed statistic exceeds the critical value, the test confirms that the proposed XG-RVFL model exhibits a statistically significant performance compared to the baseline models. To further examine these differences, we perform the Nemenyi post-hoc test, which evaluates pairwise performance variations among models. This test determines whether the rank difference between two models exceeds a predefined threshold, known as the critical difference $(C.D.)$. If the difference between their mean ranks surpasses $C.D.$, the model with the lower average rank is deemed statistically superior because rank 1 denotes the best performance. The $C.D.$ value is computed as $C.D. = q_\alpha\sqrt{\frac{l(l+1)}{6\mathcal{D}}}$, where q_α is the critical

value obtained from the studentized range statistic, adjusted by $\sqrt{2}$ for the two-tailed Nemenyi test at a chosen significance level. At 5% significance level, the value of $C.D.$ is computed as 1.97. We analyze the rank differences between the proposed XG-RVFL and the baseline models to assess statistical significance. Notably, RVFL (3.92), RVFLwoDL (4.18), IF-RVFL (2.90), and Total-Var-RVFL (1.76, close to the threshold) exhibit considerable performance gaps, indicating the superior effectiveness of the proposed XG-RVFL. In contrast, the NF-RVFL variants (NF-RVFL-K: 1.09, NF-RVFL-R: 1.19, NF-RVFL-C: 0.96) and HE-RVFL (1.54) exhibit rank differences within the $C.D.$ threshold, indicating that these differences are not statistically significant at 5% confidence level. However, considering the average accuracy, average rank, and Friedman test results, we can infer that the proposed XG-RVFL delivers superior performance over these models, demonstrating enhanced robustness and generalization across diverse binary classification tasks.

Table 7.3: Friedman test results for UCI and KEEL datasets.

Dataset type	l	$\mathcal{D}$	χ_F^2	F_F	$F((l-1),(l-1)(\mathcal{D}-1))$	**Significant difference (As per Friedman test)**
Binary UCI	9	37	78.74	13.05	1.97	Yes
Multiclass UCI	7	25	31.38	6.35	2.16	Yes
KEEL	8	24	29.71	4.94	2.07	Yes

7.1.4.2 Evaluation on UCI Multiclass Datasets

Table 7.4 presents the classification results of the proposed XG-RVFL model and baseline methods on 25 multiclass datasets, divided into two categories: M1, comprising datasets with fewer than 1,000 samples, and M2, consisting of datasets with 1,000 or more samples. For the M1 category, which includes 17 datasets, the proposed XG-RVFL model achieves the highest average accuracy of 70.0848%. Among the baseline methods, NF-RVFL-K performs the best with an accuracy of 66.8725%, followed closely by NF-RVFL-C and NF-RVFL-R with accuracies of 66.4651% and 66.4633%, respectively. Total-Var-RVFL achieves a moderate accuracy of 64.8098%, while RVFL and RVFLwoDL yield lower ac-

Table 7.4: Accuracy of the proposed XG-RVFL model compared to baseline models on each of the multiclass datasets in categories M1 and M2.

Dataset ↓ | Model →	**RVFL** [16]	**RVFLwoDL** [17]	**Total-var-RVFL** [149]	**NF-RVFL-K** [55]	**NF-RVFL-R** [55]	**NF-RVFL-C** [55]	**XG-RVFL**†
M1-Category							
audiology_std	69.3205	65.7564	50.3718	70.3718	71.3846	67.8077	66.2436
dermatology	97.5379	97.0011	97.2714	97.8193	97.5454	97.5379	97.235
ecoli	60.9175	61.2116	51.0667	60.6277	60.619	61.5145	62.0817
glass	37.6855	39.0808	34.4297	42.8239	42.3477	40.4873	55.3699
hayes_roth	61.875	60.625	62.5	61.875	61.875	65	63.125
heart_cleveland	59.7268	59.377	61.0328	61.3388	61.0383	60.0437	60.0546
heart_switzerland	45.6667	46.4667	48.9	46.5333	47.9	49.7	46.5667
heart_va	40	40	40.5	42.5	41	41.5	40.5
iris	74.6667	74	74.6667	74.6667	73.3333	74.6667	82.6667
low_res_spect	87.7605	86.6285	73.9852	87.5666	87.3849	87.7605	82.3012
lymphography	86.4138	85.0345	86.4138	87.1034	85.7701	88.5057	84.3448
pittsburg_bridges_MATERIAL	75.1948	77.013	80.6494	75.9307	77.6623	76.9697	82.9221
pittsburg_bridges_REL_L	62.1429	63.1429	64.1429	69.0476	67.2381	62.1905	67.9524
pittsburg_bridges_TYPE	42.8571	42.8571	43.8095	44.7619	45.7143	44.7619	51.9048
seeds	89.0476	87.1429	90	90	90.4762	90.4762	91.9048
synthetic_control	42.1667	42.5	57.8333	54.8333	51.1667	55.5	70.3333
vertebral_column_3clases	65.1613	65.1613	84.1935	69.0323	67.4194	65.4839	85.9355
Average Accuracy	64.5965	64.294	64.8098	<u>66.8725</u>	66.4633	66.4651	**70.0848**
M2-Category							
car	72.6265	72.1057	72.6234	70.8337	70.0139	71.0626	72.8506
contrac	40.458	40.1183	41.3386	42.3673	41.4934	41.5416	70.8707
image_segmentation	87.6623	87.0996	88.3117	83.5931	83.2035	83.6797	73.4199
led_display	72.6	72.5	73.5	72.2	72.7	73.4	67.3
molec_biol_splice	52.6959	51.8809	54.2633	68.652	69.2163	68.3072	78.9969
nursery	70.3935	70.1775	63.0093	66.3889	66.7824	71.9444	66.6667
wine_quality_red	59.4769	59.4755	59.9134	60.8533	60.4156	60.5388	78.3333
wine_quality_white	52.3699	52.5332	52.472	52.6349	52.4106	53.4925	57.9767
Average Accuracy	63.5354	63.2363	63.179	64.6904	64.5295	<u>65.4959</u>	**70.8019**
Overall Accuracy	64.257	63.9556	64.2879	<u>66.1742</u>	65.8444	66.1549	**70.3143**
Overall Rank	4.98	5.58	4.16	3.42	3.86	<u>3.1</u>	**2.9**

Here, bold indicates the best, underlined marks the second-best, and † denotes the proposed model.

curacies of 64.5965% and 64.294%, respectively, highlighting the comparative underperformance of traditional models in small-scale multiclass scenarios. In the M2 category, which comprises 8 larger datasets, the XG-RVFL again secures the highest average accuracy of 70.8019%. The second-best performance in this category is achieved by NF-RVFL-C with 65.4959%, followed by NF-RVFL-K at 64.6904% and NF-RVFL-R at 64.5295%. Total-Var-RVFL achieves 63.179%, while RVFL and RVFLwoDL register the lowest accuracies of 63.5354% and 63.2363%, respectively. The sharp improvement in XG-RVFL performance in M2 indicates its ability to scale and generalize well across larger multiclass datasets. Considering the overall performance across all multiclass datasets, the proposed XG-RVFL model achieves the highest average accuracy of 70.3143%, which is significantly higher than all baseline models. NF-RVFL-K achieves second best result with overall accuracy of 66.1742%, followed closely by NF-RVFL-C (66.1549%) and NF-RVFL-R (65.8444%). Total-Var-RVFL shows moderate performance at 64.2879%, while RVFL and

RVFLwoDL yield the lowest overall accuracies of 64.257% and 63.9556%, respectively. These results demonstrate that the XG-RVFL model consistently outperforms all compared methods in both M1 and M2 dataset categories, validating its robustness and adaptability for multiclass classification. The superior performance can be attributed to the flexibility of the proposed XG loss function, which allows dynamic control over asymmetry and boundedness, enhancing the model's capability to manage complex class distributions and noisy data environments effectively. In addition to accuracy, the overall average rank provides further insights into model consistency across datasets. The proposed XG-RVFL achieves the best average rank of 2.90, reinforcing its stability and superior generalization. NF-RVFL-C follows closely with a rank of 3.10, while NF-RVFL-K and NF-RVFL-R rank 3.42 and 3.86, respectively. Total-Var-RVFL holds a moderate rank of 4.16, whereas RVFL and RVFLwoDL exhibit the weakest performance with ranks of 4.98 and 5.58, respectively. The ranking distribution clearly demonstrates that XG-RVFL consistently outperforms all baseline models, reinforcing its effectiveness in addressing the limitations of traditional RVFL frameworks.

To assess statistical significance, the Friedman test is conducted, yielding F_F statistic of 6.35, surpassing the critical value of 2.16 at 5% significance level (see to Table 7.3). This confirms that the observed performance differences among the models are statistically significant. Further, for pairwise comparison, we employ the Nemenyi post-hoc test. For the multiclass datasets, the value of $C.D.$ at 5% significance level is computed as 1.80. The rank differences between the baseline models and the proposed XG-RVFL are as follows: RVFL (2.08), RVFLwoDL (2.68), Total-Var-RVFL (1.26), NF-RVFL-K (0.52), NF-RVFL-R (0.96), and NF-RVFL-C (0.2). As per the Nemenyi post-hoc test, RVFL and RVFLwoDL exhibit rank differences exceeding the $C.D.$, establishing that their performance is significantly lower than XG-RVFL. While the rank differences for Total-Var-RVFL and NF-RVFL

variants remain within the $C.D.$ threshold, their overall performance–considering average accuracy, average rank, and the Friedman test–indicates that the proposed XG-RVFL consistently outperforms these baseline models, reinforcing its superiority in multiclass classification.

Table 7.5: F1-Score of the proposed XG-RVFL model compared to baseline models on each of the KEEL datasets.

Dataset ↓ | **Model** →	**RVFL** [16]	**RVFLwoDL** [17]	**IF-RVFL** [101]	**NF-RVFL-K** [55]	**NF-RVFL-R** [55]	**NF-RVFL-C** [55]	**HE-RVFL** [143]	**XG-RVFL**†
KEEL Datasets								
bupa or liver-disorders	65.3742	67.4841	66.6265	65.9494	64.4552	67.4004	64.1193	66.1193
crossplane150	72.2791	72.2739	71.3304	72.0591	71.6248	72.3757	69.7421	71.7421
ecoli-0-1-4-6_vs_5	84.7418	83.6627	77.2521	74.2857	84.7778	86.1413	83.6667	86.6667
ecoli-0-1-4-7_vs_5-6	83.6694	83.3975	86.603	70.1818	83.1429	84.5275	82.5051	85.5051
ecoli-0-1_vs_2-3-5	87.8273	89.0292	88.658	68.3232	85.4902	87.8788	86.9394	89.9394
ecoli-0-1_vs_5	88.1375	90.3456	89.3232	88.1429	89.1429	82	86.6429	90.1429
ecoli-0-2-3-4_vs_5	85.8136	84.1093	82.7778	80.9524	82.2929	86.1111	86.1104	89.6104
ecoli-0-2-6-7_vs_3-5	83.1824	82.8387	84.1818	79.9524	81	82.4762	81.9286	84.4286
ecoli-0-3-4-6_vs_5	81.0697	80.0698	81.9596	77.7949	80.0067	73.1025	76.6429	79.1429
ecoli-0-3-4-7_vs_5-6	83.5375	83.5361	83.9025	80.0488	83.3333	81.3675	79.7807	82.2807
ecoli-0-4-6_vs_5	87.9162	87.9182	92.9206	88.6984	92.1558	89.2929	86.5	89
ecoli-0-6-7_vs_3-5	76.9629	77.2634	76.6667	76	69.7442	79.4286	75.5952	78.0952
ecoli-0-6-7_vs_5	85.8743	83.6521	80.3247	86.4762	81.6558	84.7778	81.3312	83.8312
ecoli01vs5	94.2284	94.2215	90	93	93.3333	95.1818	92	95
haberman	84.8527	84.8582	83.8211	83.0547	84.5914	84.0019	82.1593	84.6593
heart-stat	83.8029	83.2546	81.7728	84.9066	83.1575	83.2638	80.7018	83.2018
led7digit-0-2-4-5-6-7-8-9_vs_1	76.3251	78.8854	79.5079	78.6494	78.8398	78.0806	76.275	78.775
monk3	48.2078	46.6832	46.6687	27.7543	36.3706	28.0655	50.0322	48.5322
new-thyroid1	88.8173	88.8187	87.1292	87.7778	87.7778	87.1429	84.1193	87.1193
shuttle-6_vs_2-3	100.0049	100.0042	100	100	100	100	96.5	100
votes	92.3216	91.9998	94.667	96.0335	95.6165	95.3734	92.2932	95.7932
yeast-0-3-5-9_vs_7-8	36.0784	36.0764	42.0977	37.3605	31.7705	35.9437	41.7308	40.2308
yeast-0-5-6-7-9_vs_4	59.0625	54.8186	62.3182	61.5951	56	47.5879	66.8513	68.8513
yeast2vs8	71.8493	66.6789	73.3074	69.3651	73.0952	62.9437	74.7619	76.7619
Average F1-Score	79.2474	78.8283	<u>79.3257</u>	76.1818	77.8906	77.2694	78.2887	**80.6429**
Average Rank	<u>3.83</u>	3.88	4.08	5.4	5.19	4.46	6.17	**3**

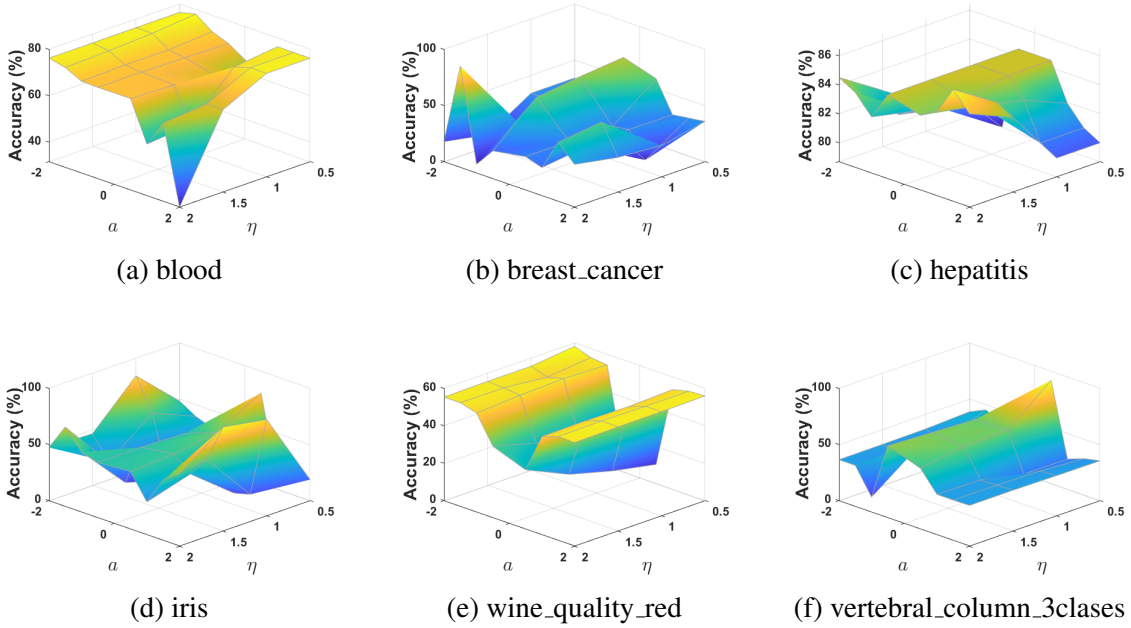
Here, bold indicates the best, underlined marks the second-best, and † denotes the proposed model.

7.1.4.3 Evaluation on KEEL Datasets

To further assess the effectiveness of the proposed XG-RVFL model in handling class-imbalanced data, we employ a diverse set of benchmark datasets from the KEEL repository. Since the KEEL datasets are highly imbalanced, the F1-score serves as a more informative metric than accuracy, as it captures the trade-off between precision and recall. Table 7.5 presents the F1-scores of the proposed XG-RVFL and baseline variants. The proposed XG-RVFL achieves the highest average F1-score of 80.64%, outperforming all baseline models. The second-best performance is obtained by IF-RVFL (79.33%), followed closely by the standard RVFL (79.25%) and RVFLwoDL (78.83%). The performance gain of XG-RVFL demonstrates its robustness and adaptability in imbalanced scenarios, benefiting from the

proposed fleXi guardian loss formulation. Although XG-RVFL exhibits consistent superiority across most datasets, a few cases show lower performance compared to NF-RVFL-C or IF-RVFL. This may be attributed to the non-convexity of the underlying optimization landscape of the fleXi guardian loss, which could cause the optimization process to converge to local minima for certain data distributions. In terms of average rank, XG-RVFL achieves the best score of 3.0, followed by the standard RVFL (3.83) and RVFLwoDL (3.88). This ranking further confirms that the proposed model consistently performs well across diverse imbalanced datasets, highlighting its reliability and generalization capability. Overall, the results affirm that XG-RVFL provides a balanced trade-off between precision and recall, outperforming both traditional and recent RVFL variants in class-imbalanced settings. To further validate the statistical reliability of the obtained results, a Friedman test is performed across all competing models on the KEEL datasets. The resulting Friedman statistic, $F_F = 4.94$, exceeds the critical threshold of 2.07 at a 5% significance level (refer to Table 7.3), thereby confirming that the observed differences in performance are not due to random chance. This establishes a statistically significant variation among the compared models. To identify which specific models differ from one another, the Nemenyi post-hoc analysis is subsequently conducted. The critical difference ($C.D.$) for the KEEL datasets, computed at a 5% significance level, is found to be 2.14. The absolute rank differences between the proposed XG-RVFL and the competing approaches are 0.83, 0.88, 1.08, 2.40, 2.19, 1.46, and 3.17 for RVFL, RVFLwoDL, IF-RVFL, NF-RVFL-K, NF-RVFL-R, NF-RVFL-C, and HE-RVFL, respectively. Among these, the rank differences for NF-RVFL-K, NF-RVFL-R, and HE-RVFL clearly exceed the computed $C.D.$, indicating that XG-RVFL achieves a statistically superior performance over these baselines. For the remaining models, RVFL, RVFLwoDL, IF-RVFL, and NF-RVFL-C, the rank gaps fall below the $C.D.$ threshold, implying that while XG-RVFL still demonstrates lower mean ranks, the improvement

is not statistically significant at the 5% level. This analysis highlights that the proposed method not only achieves the best overall rank but also provides statistically meaningful gains over several recent RVFL variants, reinforcing the robustness and reliability of its improvements across diverse imbalanced datasets.



(a) blood (b) breast_cancer (c) hepatitis

(d) iris (e) wine_quality_red (f) vertebral_column_3clases

Figure 7.2: Sensitivity analysis of the XG-RVFL model with respect to the loss hyperparameters a and η, presented through 3D surface plots for six datasets: (a) blood, (b) breast_cancer, (c) hepatitis, (d) iris, (e) wine_quality_red, and (f) vertebral_column_3clases. The plots illustrate the effect of tuning a and η on model accuracy while keeping other hyperparameters fixed at their optimal values, highlighting the significance of the loss hyperparameters in achieving robust performance.

7.1.4.4 Sensitivity Analysis

To examine the role of the proposed fleXi Guardian (XG) loss in influencing model robustness and generalization, we perform a comprehensive sensitivity analysis with respect to its hyperparameters a and η. These parameters respectively govern the degree of asymmetry and the upper bound of the loss function. The objective is to study the effects of these hyperparameters on classification accuracy while keeping all other model parameters fixed

to their optimal values, as identified through cross-validation. We conduct this analysis on six representative datasets, carefully selected to cover both binary and multiclass classification scenarios. The binary datasets include blood, breast_cancer, and hepatitis, while the multiclass datasets include iris, wine_quality_red, and vertebral_column_3clases, offering diverse challenges such as varying numbers of classes and feature spaces. For each dataset, a 3D surface plot (see Figure 7.2) illustrates the impact of varying $a \in [-2 : 0.5 : 2] \setminus \{0\}$ and $\eta \in [0.5 : 0.5 : 2]$ on the average test accuracy. Among the six datasets, distinct patterns emerge in how the accuracy responds to variations in a and η. In the blood dataset (Figure 7.2(a)), the surface shows a well-defined peak when a lies in the positive range (approximately 1.5) and η is moderate (around 1). This indicates that emphasizing right-side residuals via positive asymmetry, in combination with moderate bounding, improves generalization. A similar trend is observed in the hepatitis dataset (Figure 7.2(c)), where the model exhibits strong sensitivity to both hyperparameters: performance rises sharply within a narrow range and drops quickly outside it. This highlights that effective tuning of both asymmetry and boundedness is crucial for this data. On the other hand, the breast_cancer dataset (Figure 7.2(b)) shows relatively flat accuracy with respect to a, but a noticeable increase with higher η. This suggests that for breast_cancer, the control over the upper bound of the loss plays a more significant role than directional penalization. For the iris dataset (Figure 7.2(d)), accuracy is consistently high for most values of η, especially when $\eta > 1$, and remains nearly invariant across a. This implies that symmetric loss behavior suffices, and only moderate bounding is needed for high performance, likely due to the simplicity and separability of this dataset. The wine_quality_red dataset (Figure 7.2(e)) shows a mild but consistent increase in accuracy with increasing η, with minimal sensitivity to a, suggesting that loss boundedness drives performance, whereas asymmetry has negligible effect. In contrast, the vertebral_column_3clases dataset (Figure 7.2(f)) presents a sharp and narrow

ridge of optimal accuracy–a clear interaction effect between a and η. Here, both parameters must be carefully balanced; deviating from this optimal zone causes a rapid drop in accuracy, indicating the dataset's sensitivity to misclassification direction and scale. Overall, this analysis highlights the expressive flexibility of the proposed XG loss, which enables the XG-RVFL model to adapt its behavior based on the structure and difficulty of the data. The parameters a and η serve as powerful levers to balance asymmetry and boundedness in the loss, directly influencing the model's robustness and generalization. The 3D surface plots confirm that while the loss function can be fine-tuned for optimal performance, its sensitivity varies with the data characteristics–indicating that informed selection of a and η is not only beneficial but often essential for achieving reliable classification results.

7.2 Wave-BLS Framework

7.2.1 Background and Notation

This section fixes the notation needed for Wave-BLS. Let $X \in \mathbb{R}^{m\times d}$ be the input matrix and $Y \in \mathbb{R}^{m\times c}$ the target matrix. In BLS [19], the feature-layer output is denoted by $Z \in \mathbb{R}^{m\times pq}$, where p is the number of nodes per feature window and q is the number of feature windows. The enhancement-layer output is denoted by $H \in \mathbb{R}^{m\times rs}$, where r is the number of nodes per enhancement window and s is the number of enhancement windows. The concatenated broad representation is $A = [Z, H] \in \mathbb{R}^{m\times(pq+rs)}$, and $W \in \mathbb{R}^{(pq+rs)\times c}$ denotes the output weight matrix.

Standard BLS learns W by least-squares fitting on $AW - Y$. This maintains the computational appeal of broad randomized representations, but the squared-error term remains sensitive to noisy observations and outliers. Wave-BLS therefore keeps the broad representation A and replaces the least-squares loss by the bounded and smooth Wave loss, followed

by an inversion-free Nesterov accelerated gradient optimization scheme.

7.2.2 Proposed Wave-BLS Framework

In this section, we introduce Wave-BLS, i.e., the broad learning system based on the wave loss function. We first discuss the wave loss function and provide its key characteristics. Subsequently, we detail the formulation and optimization of the proposed Wave-BLS model.

7.2.2.1 Wave Loss Function

The wave loss function [67] is a recently introduced asymmetric, bounded, and smooth loss designed to provide controlled error penalization. It is defined as

$$L_{\text{wave}}(u) = \frac{u^2 e^{au}}{1 + \lambda u^2 e^{au}}, \tag{7.10}$$

where $u \in \mathbb{R}$ denotes the local prediction-error variable used in the Wave-BLS loss, $a \in \mathbb{R}$ is a shape parameter governing the asymmetry and growth behavior of the loss, and $\lambda \in \mathbb{R}^+$ is a bounding parameter that limits the maximum penalty induced by large errors. This local u is distinct from the classification margin-violation variable $u_i = 1 - y_i f(\mathbf{x}_i)$ defined in Chapter 2. Fig. 7.3 illustrates the behavior of the wave loss function under different values of a and λ.

The key properties of the wave loss function, which are directly leveraged in the proposed Wave-BLS framework, are summarized below.

- **Asymmetry:** The wave loss is asymmetric, enabling unequal penalization of overestimation and underestimation errors through the shape parameter a. Positive values of a emphasize overpredictions, while negative values accentuate underpredictions; the loss becomes symmetric when $a = 0$.

- **Boundedness:** The wave loss is upper-bounded by $1/\lambda$, ensuring that large errors exert limited influence on the optimization process. This bounded nature prevents excessive domination by outliers and enhances robustness in noisy learning environments.

- **Smoothness:** The wave loss is continuously differentiable, facilitating efficient and stable optimization. This smoothness enables the use of efficient gradient-based solvers.

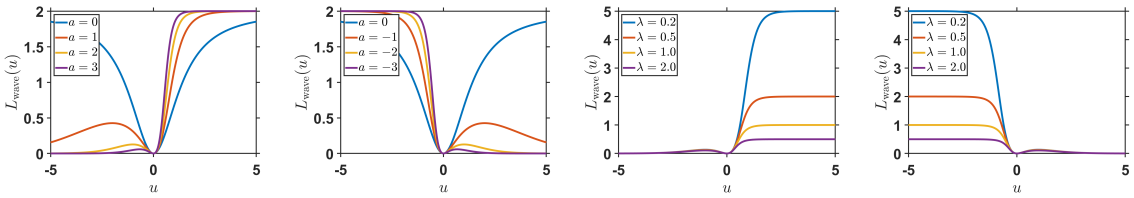


Figure 7.3: Illustration of wave loss function, showing effect of asymmetry parameter a (positive, negative, and $a = 0$) and the bounding parameter λ.

These properties enable the wave loss function to address the fundamental issues of traditional loss functions.

7.2.2.2 Formulation & Optimization: Broad Learning System with Wave Loss Function (Wave-BLS)

In this subsection, we incorporate the wave loss function into the BLS by replacing the conventional squared error loss, resulting in a robust learning framework termed broad learning system with wave loss function, i.e., Wave-BLS. The proposed formulation improves robustness against noise and outliers while enabling scalable optimization by avoiding explicit matrix inversion during parameter estimation.

The optimization problem of Wave-BLS is formulated as

$$J(W) = \frac{1}{2}\|W\|_F^2 + C\, L_{\text{wave}}(AW - Y), \tag{7.11}$$

where $C > 0$ is the regularization parameter controlling the trade-off between model complexity and empirical risk, and $L_{\text{wave}}(AW - Y)$ represents the aggregated wave loss over all training samples.

The wave loss for the ith sample and j^{th} class is defined as

$$L_{\text{wave}}(u_{ij}) = \frac{u_{ij}^2 e^{au_{ij}}}{1 + \lambda u_{ij}^2 e^{au_{ij}}}, \tag{7.12}$$

where $u_{ij} = (AW)_{ij} - Y_{ij}$ denotes the prediction error for sample i and class j.

Accordingly, the total wave loss over the training set is computed as

$$L_{\text{wave}}(AW - Y) = \sum_{i=1}^{m} \sum_{j=1}^{c} L_{\text{wave}}(u_{ij}). \tag{7.13}$$

Hence, the objective function in (7.11) can be equivalently rewritten as

$$J(W) = \frac{1}{2}\|W\|_F^2 + C \sum_{i=1}^{m} \sum_{j=1}^{c} L_{\text{wave}}(u_{ij}). \tag{7.14}$$

The objective function in (7.14) consists of two components:

1. **Regularization term** $\left(\frac{1}{2}\|W\|_F^2\right)$, which controls model complexity and promotes generalization in accordance with the principle of structural risk minimization.

2. **Empirical loss term** $\left(\sum_{i=1}^{m} \sum_{j=1}^{c} L_{\text{wave}}(u_{ij})\right)$, which aggregates the wave loss over all samples and classes, thereby enhancing robustness through its asymmetric, bounded, and smooth characteristics.

To efficiently optimize the objective function in (7.14), we exploit its smoothness and use a gradient-based optimization strategy. This approach eliminates the need for explicit matrix inversion, which is a computational bottleneck in conventional BLS, thereby significantly improving scalability.

Specifically, we adopt the Nesterov accelerated gradient (NAG) algorithm due to its fast convergence and stable update behavior. The look-ahead mechanism of NAG enables accurate gradient estimates, while its momentum-based formulation effectively suppresses oscillations in ill-conditioned optimization landscapes.

Further, to ensure stable and efficient convergence, the learning rate is dynamically adjusted using an exponential decay schedule. At iteration t, the learning rate is updated as $\mu_t = \mu_0\, e^{-\eta t}$, where μ_0 is the initial learning rate and η controls the decay rate. This adaptive strategy balances convergence speed and stability, leading to reliable optimization of the proposed Wave-BLS model.

To optimize the objective function in (7.14), we compute its gradient with respect to the output weight matrix W. The resulting gradient consists of contributions from the regularization term and the wave loss term and is used within the NAG-based optimization scheme.

For a given class $j \in \{1, 2, \ldots, c\}$, the gradient of the wave loss term with respect to $W(:, j)$ is given by

$$\nabla_{W(:,j)} L_{\text{wave}} = \sum_{i=1}^{m} \frac{u_{ij}\big(2 + a u_{ij}\big) e^{a u_{ij}}}{\big(1 + \lambda u_{ij}^2 e^{a u_{ij}}\big)^2}\, A(i,:)^\top. \tag{7.15}$$

This gradient is subsequently used to update the model parameters within the NAG optimization framework.

The overall gradient of the objective function $J(W)$ is given by

$$\nabla J(W) = W + C\, \nabla_W L_{\text{wave}}, \tag{7.16}$$

where the first term arises from the regularization component $\frac{1}{2}\|W\|_F^2$, and the second term corresponds to the contribution of the wave loss.

Algorithm 7.3 NAG-based optimization for Wave-BLS

Input: $A \in \mathbb{R}^{m\times(pq+rs)}$: representation matrix, $Y \in \mathbb{R}^{m\times c}$: target matrix, C: regularization parameter, $I_{\max}$: maximum number of iterations, $\mu^{(0)}$: initial learning rate, γ: momentum coefficient, η: learning rate decay factor, δ: convergence tolerance.
Output: $W \in \mathbb{R}^{(pq+rs)\times c}$: output weight matrix.

1: Initialize $t \leftarrow 0$, $W^{(0)}$, $v^{(0)} \leftarrow \mathbf{0}$, $\mu^{(0)}$.
2: **repeat**
3: Compute: $W^{\text{look-ahead}} = W^{(t)} + \gamma v^{(t)}$.
4: Compute stochastic gradient $g^{(t)} = \nabla J(W^{\text{look-ahead}})$.
5: Velocity update:

$$v^{(t+1)} = \gamma v^{(t)} - \mu^{(t)} g^{(t)}.$$

6: Model parameters update:

$$W^{(t+1)} = W^{(t)} + v^{(t+1)}.$$

7: Learning rate update:

$$\mu^{(t+1)} = \mu^{(t)} e^{-\eta t}.$$

8: $t \leftarrow t + 1$.
9: **until** $t \geq I_{\max}$ **or** $\|W^{(t)} - W^{(t-1)}\|_F \leq \delta$
10: **return** W.

At the tth iteration, the gradient is computed using the entire training dataset, ensuring stable and deterministic updates. Algorithm 7.3 provides the complete procedure for computing the output weight matrix W in the proposed Wave-BLS model using the NAG-based optimization scheme.

The conditional computational trade-off for Wave-BLS and XG-RVFL is discussed in Section 7.5.

7.2.3 Wave-BLS Experiments and Discussion

This section empirically evaluates the proposed Wave-BLS against baseline models on 30 UCI benchmark datasets under both clean and noisy conditions. The proposed Wave-BLS model is evaluated against several baseline methods, including RVFL [16], ELM, i.e., RVFL without direct link(RVFLwoDL) [17], BLS [19], Wave-RVFL [150], NF-BLS [151], F-BLS [152], IF-BLS [152], and KRP-BLS [153]. The experimental setup is described

below.

7.2.3.1 Experimental Setup

All experiments are conducted using MATLAB R2023a on a workstation equipped with an 11th-generation Intel Core i7-11700 processor running at 2.50 GHz, 16 GB of RAM, and a Windows 11 operating system. To ensure a fair and reliable comparison, a 5-fold cross-validation strategy is employed in conjunction with grid search for hyperparameter tuning. Specifically, each dataset is partitioned into five mutually exclusive folds. For a given set of hyperparameters, the model is trained on four folds and evaluated on the remaining fold, repeating this process until each fold has served as the test set once. The testing accuracy obtained across the five folds is then averaged, and the highest average accuracy achieved over all hyperparameter configurations is reported as the final performance of each model.

For all models, the regularization parameter C is selected from the range $\{10^{-6}, 10^{-4}, \ldots, 10^{6}\}$. For RVFL and ELM, the number of hidden nodes is tuned within the range $5{:}10{:}205$. For BLS and the proposed Wave-BLS, the number of feature node windows is chosen from $1{:}2{:}21$, the number of feature nodes per window from $5{:}5{:}50$, and the number of enhancement nodes from $5{:}10{:}105$. For NF-BLS, the number of fuzzy groups is selected from $1{:}2{:}21$, the number of fuzzy nodes per group from $5{:}5{:}50$, and the number of enhancement nodes from $5{:}10{:}105$. For IF-RVFL, the kernel hyperparameter is selected from the range $\{10^{-6}, 10^{-4}, \ldots, 10^{6}\}$. For Wave-RVFL, hyperparameter settings are adopted from [150] to ensure a fair and consistent comparison. Similarly, the configurations for F-BLS and IF-BLS follow the settings reported in [152], while the parameters for KRP-BLS are set according to [153], maintaining consistency with their original implementations. For the proposed Wave-BLS model, the wave loss parameters are selected from $a \in \{-5, -4, \ldots, 5\}$ and $\lambda \in \{0.1, 0.5, 1\}$. The parameters of the NAG-based optimization

algorithm are fixed as follows: the initial weight matrix is set to $W^{(0)} = 0.01 \times \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{(pq+rs)\times c}$ denotes a matrix of ones; the initial velocity is initialized as $v^{(0)} = \mathbf{0}$; the initial learning rate is set to $\mu^{(0)} = 0.01$; the learning rate decay factor is fixed to $\eta = 0.1$; the momentum coefficient is set to $\gamma = 0.6$; the convergence tolerance is set to $\delta = 10^{-6}$; and the maximum number of iterations is fixed to $I_{\max} = 100$.

Table 7.6 reports the dataset-wise accuracy values used to compare Wave-BLS with the baseline randomized neural network models.

Table 7.6: Dataset wise results of the proposed wave-BLS model against the baseline models on 30 UCI datasets.

Dataset ↓ | Model →	**RVFL [16]**	**RVFLwoDL [17]**	**BLS [19]**	**Wave-RVFL [150]**	**NF-BLS [151]**	**F-BLS [152]**	**IF-BLS [152]**	**KRP-BLS [153]**	**Wave-BLS†**
acute_inflammation	100	100	100	100	100	100	100	100	100
acute_nephritis	100	100	100	100	100	100	100	100	100
bank	90.9051	89.4051	89.7366	88.1779	89.5817	89.7590	89.4051	87.7230	89.0202
blood	78.4056	76.9056	79.6272	80.7578	76.0807	78.5807	77.2451	76.7000	82.9184
breast_cancer	68.1727	66.6727	69.8851	70.2179	70.1754	72.2868	83.1579	82.4947	90.8772
breast_cancer_wisc	89.4897	87.9897	88.4183	86.8050	90.7081	88.2775	88.9938	89.3240	97.8571
chess_krvkp	73.5312	72.0312	84.3862	75.7371	70.4004	84.2921	84.9184	87.3718	88.7031
conn_bench_sonar_mines_rocks	62.0807	60.5807	69.2451	63.9431	60.6272	69.1521	80.2091	78.2049	93.6585
credit_approval	86.8623	85.3623	87.5362	88.5995	84.4928	86.3768	88.5507	89.7797	87.1014
cylinder_bands	67.7193	66.2193	69.9258	69.7509	69.5336	69.1338	72.8536	70.8109	70.3008
echocardiogram	85.4031	83.9031	83.9316	87.9652	80.9402	84.6724	88.4900	86.7202	86.2108
fertility	89.5000	90.0000	90.0000	86.8150	92.0000	91.0000	91.0000	89.1800	92.0000
haberman_survival	74.9902	73.4902	70.2750	77.2399	73.4902	69.6563	75.4574	76.1937	78.7361
hepatitis	83.2258	83.2258	85.1613	85.7226	87.7419	84.5161	87.7419	85.1096	85.2258
hill_valley	77.9706	77.9706	82.0185	80.3097	78.9583	81.6002	79.6249	78.6344	80.4570
horse_colic	84.2098	84.7098	86.1422	86.7361	83.9726	86.1385	86.6938	84.0920	85.7919
mammographic	79.0889	79.0889	78.6712	81.4616	79.5040	78.6744	79.8192	79.8238	84.2870
molec_biol_promoter	68.9610	68.9610	83.9394	71.0298	75.1515	84.9351	88.7879	89.1241	96.1905
monks_1	83.2497	83.2497	75.3314	85.7472	86.1277	77.1396	77.6705	74.0404	74.1203
musk_1	67.8640	67.8640	75.8311	69.8999	67.8487	76.6754	78.9846	79.8523	96.6316
oocytes_merluccius_nucleus_4d	79.8407	79.8407	82.7776	82.2359	80.1420	81.8011	80.9201	79.3017	72.8541
oocytes_trisopterus_nucleus_2f	76.4211	76.4211	78.9371	78.7137	75.9911	78.6123	75.9899	74.2302	76.2815
pima	71.4888	71.4888	72.0958	73.6335	72.7918	71.4846	73.3079	74.1087	76.4341
pittsburg_bridges_T_OR_D	87.6429	88.1429	89.1095	90.2722	90.1429	88.1905	90.2381	87.0791	92.1429
spambase	87.1125	87.1125	89.0201	89.7259	83.2657	89.4582	90.7407	87.4741	93.9201
spect	67.9245	67.9245	69.4340	69.9622	67.9245	69.4340	72.4528	72.3517	73.9623
statlog_heart	80.0000	80.0000	82.2222	82.4000	80.7407	82.2222	84.0741	85.0566	87.0370
tic_tac_toe	86.0068	86.0068	98.4315	88.5870	82.7623	97.8081	97.0768	94.1645	99.0201
titanic	77.9168	77.9168	77.9168	80.2543	78.0537	77.9168	79.9532	78.3541	94.6875
vertebral_column_2clases	70.6452	70.6452	72.2453	72.7646	72.0465	71.3333	72.9479	72.7590	75.9168
Average Accuracy	79.8876	79.4376	82.0751	81.5155	80.0399	82.0376	<u>83.9102</u>	83.002	**86.7448**
Average Rank	6.5333	7.1833	4.8	4.0667	6.0833	5.2667	<u>3.4833</u>	4.8333	**2.75**

† indicates the proposed model; **bold** denotes the best performance, and <u>underline</u> denotes the second-best.

Table 7.7: Robustness comparison of BLS and Wave-BLS under outlier and noise contamination.

Dataset	Level	Outliers		Level	Noise	
		BLS	Wave-BLS†		BLS	Wave-BLS†
blood	Clean	79.6272	**82.9184**	Clean	79.6272	**82.9184**
	5%	75.8421	**80.4067**	5%	77.1034	**81.7749**
	10%	72.5186	**78.3925**	10%	74.8652	**79.9368**
	15%	69.2847	**75.8106**	15%	71.4928	**77.1053**
	20%	66.0319	**72.4678**	20%	68.1074	**74.6389**
	Avg.	72.6609	**78.3992**	Avg.	74.3398	**79.2749**
horse_colic	Clean	**86.1422**	85.7919	Clean	**86.1422**	85.7919
	5%	82.6374	**84.9563**	5%	83.9046	**85.1187**
	10%	78.9421	**82.4018**	10%	80.2873	**82.9365**
	15%	75.6189	**79.6674**	15%	76.9548	**80.3412**
	20%	72.3046	**76.1897**	20%	73.6285	**77.5403**
	Avg.	79.1290	**81.8014**	Avg.	80.1835	**82.3457**
Overall Average		75.8949	**80.1003**		77.2617	**80.8103**
Average Rank		2.0	**1.0**		1.75	**1.25**

Bold values denote the best performance for each row.

7.2.3.2 Robustness Analysis under Noise and Outliers

To evaluate the robustness of the proposed Wave-BLS, we conduct controlled experiments under outlier and noise contamination on two representative datasets. Specifically, artificial perturbations are introduced by contaminating 5%, 10%, 15%, and 20% of the training samples with outliers and label noise, respectively. The blood dataset is selected as a case where Wave-BLS consistently outperforms the standard BLS, while the horse_colic dataset represents a complementary scenario where BLS exhibits slightly better performance on the clean data. This selection enables a balanced and unbiased assessment of robustness behavior. As reported in Table 7.7, Wave-BLS demonstrates clear robustness advantages on the blood dataset across all contamination levels. Under both outlier and noise perturbations, Wave-BLS maintains higher classification accuracy than BLS, with performance degradation occurring at a noticeably slower rate as contamination increases. This behavior highlights the effectiveness of the wave loss in bounding the influence of corrupted samples and mitigating the impact of extreme deviations. For the horse_colic dataset, although BLS achieves marginally higher accuracy on the clean data, Wave-BLS consistently surpasses BLS once outliers or noise are introduced. As contamination levels increase, Wave-BLS preserves stronger performance stability, leading to higher average accuracy under both outlier and noise settings. This indicates that even in cases where the baseline model performs well on clean data, the proposed wave loss enhances resilience under adverse conditions. Overall, Wave-BLS attains superior average accuracy and lower average rank across both datasets and contamination types. These results confirm that the proposed Wave-BLS framework offers improved robustness to data corruption, making it more suitable for real-world learning scenarios where noise and outliers are unavoidable.

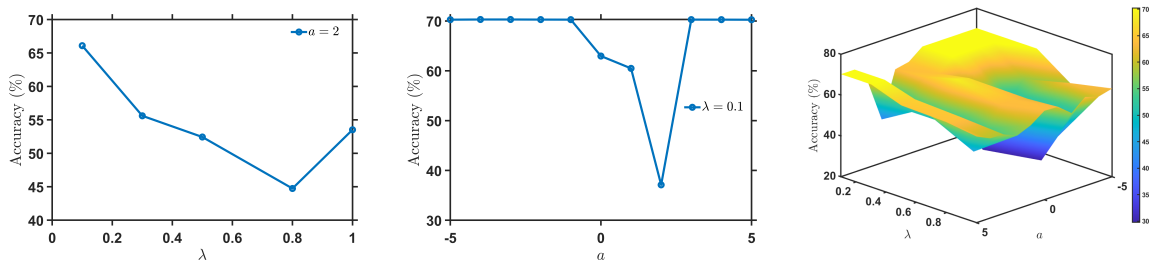
7.3 Statistical Significance Analysis

To assess whether the observed differences in average ranks among the competing models are statistically significant, we first employ the nonparametric Friedman test. Let $\mathcal{K}$ denote the number of evaluated models and $\mathcal{N}$ denote the number of datasets. This test examines the null hypothesis that all $\mathcal{K}$ models exhibit equivalent performance across $\mathcal{N}$ datasets. The Friedman test statistic is defined as $\chi_F^2 = \frac{12\mathcal{N}}{\mathcal{K}(\mathcal{K}+1)} \left(\sum_{k=1}^{\mathcal{K}} \varrho(k, \bullet)^2 - \frac{\mathcal{K}(\mathcal{K}+1)^2}{4} \right)$, where $\varrho(k, \bullet)$ denotes the average rank of the kth model. In the test, the statistic $F_F = \chi_F^2 \left(\frac{\mathcal{N}-1}{\mathcal{N}(\mathcal{K}-1)-\chi_F^2} \right)$ is commonly used, which follows an F-distribution with $(\mathcal{K}-1)$ and $(\mathcal{K}-1)(\mathcal{N}-1)$ degrees of freedom. In our study, $\mathcal{K} = 9$ models are evaluated on $\mathcal{N} = 30$ UCI benchmark datasets. As reported in Table 7.8, the Friedman statistic is $\chi_F^2 = 66.6578$, yielding a corrected value of $F_F = 11.1518$. The corresponding critical value at the 5% significance level is $F(8, 232) = 1.9784$. As the computed statistic satisfies $11.1518 > 1.9784$, the null hypothesis of equivalent model performance is rejected, confirming the presence of statistically significant differences across the evaluated methods. Next, the Nemenyi post-hoc procedure is employed to conduct pairwise comparisons between the proposed Wave-BLS and the competing models. According to the Nemenyi test, two models are considered significantly different if the difference in their average ranks exceeds the critical difference (C.D.), given by C.D. $= q_\alpha \sqrt{\frac{\mathcal{K}(\mathcal{K}+1)}{6\mathcal{N}}}$, where q_α is the critical value corresponding to the two-tailed Nemenyi test. At $\alpha = 0.1$, the resulting critical difference is C.D. $= 2.0188$. The post-hoc comparison results are summarized in Table 7.8. The proposed Wave-BLS achieves the best average rank of 2.75 and shows statistically significant improvements over RVFL, RVFLwoDL, BLS, NF-BLS, F-BLS, and KRP-BLS, as the corresponding rank differences exceed the critical difference. In contrast, the rank differences between Wave-BLS and Wave-RVFL, as well as IF-BLS, do not exceed the critical threshold, indicating com-

parable performance among these top-performing methods. Overall, the Friedman and Nemenyi tests jointly confirm that the superior performance of Wave-BLS is not due to random variation, but is statistically significant across the considered benchmark datasets.

Table 7.8: Statistical significance analysis using Friedman test and Nemenyi post-hoc test.

Friedman Test					
$\mathcal{K}$	$\mathcal{N}$	χ_F^2	F_F	$F((\mathcal{K}-1),(\mathcal{K}-1)(\mathcal{N}-1))$	Significant difference
9	30	66.6578	11.1518	1.9784	Yes
Nemenyi Post-hoc Test (C.D. = 2.0188)					
Model	Avg. rank	Rank diff.	Significant difference		
RVFL [16]	6.5333	3.7833	Yes		
RVFLwoDL [17]	7.1833	4.4333	Yes		
BLS [19]	4.8000	2.0500	Yes		
Wave-RVFL [150]	4.0667	1.3167	No		
NF-BLS [151]	6.0833	3.3333	Yes		
F-BLS [152]	5.2667	2.5167	Yes		
IF-BLS [152]	3.4833	0.7333	No		
KRP-BLS [153]	4.8333	2.0833	Yes		
Wave-BLS†	2.7500	–	N/A		



(a) (b) (c)

Figure 7.4: Sensitivity analysis of the wave loss hyperparameters in the proposed Wave-BLS model on the breast_cancer dataset. (a) Effect of the bounding parameter λ, (b) effect of the shape parameter a, and (c) their joint influence on classification accuracy.

7.4 Sensitivity Analysis of Wave Loss Hyperparameters

In this subsection, we conduct a sensitivity analysis on the breast_cancer dataset with respect to the wave loss hyperparameters. Specifically, we analyze the influence of the bounding parameter λ, the shape parameter a, and their joint interaction on classification accuracy, while fixing all remaining model hyperparameters at their optimal values. Figure 7.4(a) shows the variation in classification accuracy on the breast_cancer dataset as the bounding parameter λ is varied while fixing a. Although the exact accuracy trend may differ

across datasets, the results clearly indicate that the choice of λ critically influences model performance. In particular, intermediate values of λ lead to improved accuracy, whereas very small or large values result in performance degradation. This observation highlights the role of λ in controlling the saturation behavior of the wave loss and regulating the influence of large errors, underscoring the importance of selecting an appropriate bounding parameter for stable and robust learning. Figure 7.4(b) depicts the sensitivity of accuracy with respect to the asymmetry parameter a while keeping λ fixed. For this dataset, Wave-BLS achieves consistently high accuracy over a broad range of both negative and positive values of a, indicating that the model is not overly sensitive to the choice of the asymmetry parameter. A sharp degradation in performance is observed only at a specific intermediate value of a, suggesting that certain configurations may induce unfavorable skewness in loss penalization. Nevertheless, the rapid recovery of accuracy for neighboring values highlights that Wave-BLS remains largely stable with respect to variations in a when other hyperparameters are fixed. Figure 7.4(c) presents the joint sensitivity surface of accuracy with respect to a and λ. The surface reveals a broad region of relatively high accuracy, interspersed with localized valleys corresponding to unfavorable parameter combinations. This behavior indicates that, for this dataset, Wave-BLS does not require extremely precise tuning of a and λ to achieve competitive performance. At the same time, the presence of distinct low-accuracy regions highlights the importance of jointly selecting these parameters, as their interaction directly influences the balance between asymmetry and bounded error penalization.

Overall, this sensitivity analysis demonstrates that the wave loss hyperparameters play a meaningful role in shaping the performance of Wave-BLS. While the model exhibits a reasonable degree of stability across wide ranges of both a and λ, suboptimal parameter choices can still lead to noticeable performance degradation. These observations suggest

that a careful and informed selection of the wave loss parameters is essential to fully exploit its robustness properties and achieve consistently strong performance across datasets.

7.5 Computational Trade-Offs and Evaluation Scope

For XG-RVFL, a mini-batch update has leading cost $\mathcal{O}(kc(d+h))$, giving $\mathcal{O}(I_{\max}kc(d+h))$ over $I_{\max}$ iterations. Wave-BLS uses the complete training set at each update; if $s = pq+rs$ denotes the broad representation width, its leading iterative cost is $\mathcal{O}(I_{\max}mcs)$. These expressions avoid the cubic factorization associated with a closed-form output solution, but they also show that the gain depends on convergence. Large iteration counts, full-batch gradients, unfavorable conditioning, or conservative learning rates can offset the saving from removing explicit inversion. Conversely, large or highly redundant representation matrices can make inversion-free optimization attractive because it avoids storing and factorizing a large Gram matrix.

The experiments fix $I_{\max} = 500$ and mini-batch size $k = 64$ for XG-RVFL, and $I_{\max} = 100$ with full-batch gradients for Wave-BLS. They establish performance under these settings but do not provide a systematic runtime–accuracy study over iteration count, mini-batch size, tolerance, and conditioning. Consequently, the term *inversion-free* is an algorithmic statement, whereas the claimed scalability is conditional on the iterative solver reaching a satisfactory solution with fewer resources than the corresponding inversion-based method.

Randomized neural networks also vary with the sampled hidden parameters. The current evaluation does not report a comprehensive multi-seed distribution for every dataset and model. Cross-validation averages variation over data folds but does not isolate variation caused by the random hidden representation. The reported performance gaps should therefore be interpreted conditional on the adopted initialization procedure. A stronger as-

sessment would repeat every configuration over several fixed seeds and report the mean, standard deviation, and a paired statistical comparison against the inversion-based counterpart.

7.6 Chapter Summary

The conclusions in this summary concern the reported benchmark protocol. They should not be read as a universal runtime advantage or as evidence that performance is invariant to hidden-weight initialization; both claims require the solver trade-off and multi-seed analyses identified in Section 7.5. This chapter developed robust and inversion-free learning strategies for randomized neural networks by replacing squared-error dominated training objectives with bounded loss-based formulations. The first part introduced XG-RVFL, a computationally efficient random vector functional link model equipped with the fleXi Guardian (XG) loss. The XG loss provides boundedness, smoothness, Lipschitz continuity, and asymmetric error treatment, allowing the model to reduce the influence of noisy and outlying samples while avoiding the matrix inversion required in conventional RVFL training. The theoretical analysis, including influence-function behavior and a Rademacher-complexity-based generalization bound, supports the robustness and expected generalization of the proposed formulation.

The second part extended the same loss-design philosophy to broad learning systems through Wave-BLS. By integrating the Wave loss into the BLS framework and optimizing the resulting objective using a Nesterov accelerated gradient (NAG)-based scheme, the model preserves the efficiency of the flat BLS architecture while improving resistance to noise, outliers, and uncertain training samples. Across benchmark datasets, noise-injection experiments, and sensitivity analyses, both XG-RVFL and Wave-BLS demonstrate that robust bounded losses can substantially improve randomized neural networks without increas-

ing architectural complexity. Overall, the chapter shows that loss-function design is a practical route toward stable, scalable, and inversion-free randomized learning.

Chapter 8

CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks

This chapter introduces Copula-Aligned Weight Initialization (CAWI), a dependency-aware initialization strategy for randomized neural networks (RdNNs). RdNNs achieve efficient, backpropagation-free training by freezing randomly initialized input-to-hidden weights and solving only the output layer in closed form. However, conventional independent random initialization ignores inter-feature dependence, including correlations, asymmetries, and tail dependence, which can degrade hidden representations, conditioning, and predictive performance. CAWI addresses this limitation by fitting a copula to rank-transformed training features and sampling input-to-hidden weights from the learned dependence structure while keeping the objective, solver, and freeze-once training paradigm unchanged. The chapter develops CAWI for elliptical copulas, including Gaussian and t copulas, and Archimedean copulas, including Clayton, Frank, and Gumbel copulas. Through experiments on 83 classification benchmarks and two biomedical datasets, BreaKHis and Schizophrenia, using shallow and deep RdNN architectures, the chapter demonstrates that

dependence-aware initialization consistently improves predictive performance over conventional independent initialization.

8.1 Introduction

Deep learning (DL) models have achieved remarkable success across many domains, from image recognition [154] to natural language processing [155], due to their ability to learn rich hierarchical representations of data [14]. End-to-end training with backpropagation refines millions of parameters to minimize task-specific losses, yielding expressive features and strong generalization. However, state-of-the-art performance typically demands substantial computation, broad hyperparameter search, and long training cycles [15]. Furthermore, training deep architectures can be hindered by vanishing or exploding gradients.

The aforementioned limitations of DL models have prompted researchers to seek alternative neural architectures that couple competitive predictive performance with markedly lower training cost. A prominent family is randomized neural networks (RdNNs) [16, 140]. In RdNNs, a substantial subset of parameters–typically input-to-hidden weights–are sampled once from simple distributions and then held fixed, while the remaining (often output) weights are obtained by solving a single linear system, commonly with Tikhonov regularization, instead of iterative backpropagation [141]. This paradigm eliminates the need for iterative weight updates, substantially reducing training time and computational overhead, yet still allows RdNNs to learn complex input-output mappings [141]. Theoretically, RdNNs satisfy universal approximation on compact domains, approximating any continuous function arbitrarily well given sufficient width and appropriate activations [156].

In recent years, RdNNs have witnessed significant advancements, particularly in methods aimed at improving their robustness and scalability. Ensemble-based approaches have been introduced to enhance generalization and mitigate variability [20]. Granular ball-based

scalable methods have further extended the applicability of RdNNs to large-scale and high-dimensional datasets [57]. Additionally, the integration of fuzzy inference systems into RdNN frameworks has provided a way to incorporate uncertainty handling and interpretability, making them more effective for complex and imprecise decision-making tasks [55, 56]. Beyond these trends, robust and inversion-free learning strategies have further broadened the RdNN toolkit [157].

Despite several advancements, RdNNs face a fundamental limitation rooted in the very mechanism that makes them efficient: the input-to-hidden weights are randomly initialized once and then held fixed throughout training. With the matrix $W \in \mathbb{R}^{d\times h}$ frozen, the induced feature map $x \mapsto \phi(xW + b)$ is non-adaptive; the hidden representation $H = \phi(XW + \mathbf{1}_m b^\top) \in \mathbb{R}^{m\times h}$ is therefore determined by this random draw rather than learned from data. The fixed representation H cannot adjust to dependencies among inputs (e.g., correlations, relative scales, higher-order interactions), so at practical hidden dimension h it may underrepresent the joint patterns that drive prediction. The readout then computes $\hat{Y} = H\Theta$ with $\Theta \in \mathbb{R}^{h\times c}$, where c is the number of output classes or coordinates; if relevant dependencies are absent from H, they cannot be recovered, leading to persistent approximation bias and lower performance at a given model size.

The preceding limitation naturally raises a question: can we make the frozen projections dependence-aware without abandoning closed-form training? In this work, we answer this question by introducing CAWI (Copula-Aligned Weight Initialization), a method that uses copula theory to generate *dependence-aware* input-to-hidden weights. Copulas model multivariate data by separating the univariate marginals from the joint dependence structure, allowing us to learn dependence without committing to specific marginal forms [22, 58]. This separation is well suited to tabular learning, where feature scales and marginals vary widely but predictive signal often resides in cross-feature structure.

Several works in deep neural networks have explored structured or data-aware initialization strategies, such as orthogonal weight initialization and whitening-based transformations. Orthogonal schemes stabilize signal propagation via algebraic constraints but remain distribution-agnostic, while whitening decorrelates features through covariance normalization, primarily addressing linear dependencies. More generally, data-dependent projection methods modify the feature space (e.g., via PCA or kernel mappings), often at additional computational cost.

In contrast, CAWI is designed for RdNNs, where hidden weights are sampled once and kept fixed. Rather than altering the feature representation or imposing structural constraints, CAWI modifies only the sampling distribution of the weight matrix using an empirical copula fitted to the training features. Specifically, we construct rank-based pseudo-observations, estimate elliptical (Gaussian, t) or Archimedean (Clayton, Frank, Gumbel) copulas, and draw each column of W from the fitted dependence structure with fixed marginal scaling. The resulting frozen projections inherit empirical inter-feature dependence, while the RdNN architecture, activation functions, objective, and closed-form solver remain unchanged, incurring essentially no additional training cost.

Our Contributions

- We introduce CAWI (Copula-Aligned Weight Initialization), a plug-in scheme that makes randomized neural networks dependence-aware by sampling the columns of W from a copula fitted to the training features, while preserving the closed-form solver and the freeze-once paradigm.

- We provide numerically stable constructions for elliptical (Gaussian, t) and Archimedean (Clayton, Frank, Gumbel) copulas via rank-based estimation and nearest-SPD projection.

- We evaluate CAWI on 83 diverse classification benchmarks (binary and multiclass) and two biomedical datasets (BreaKHis and the Schizophrenia dataset) using both shallow and deep RdNN architectures, and observe consistent improvements, with 4–5% relative gains on some datasets.

8.2 Preliminaries

8.2.1 Randomized Neural Network (RdNN) Setting

Let $X \in \mathbb{R}^{m\times d}$ be the input matrix (m samples, d features) and $Y \in \mathbb{R}^{m\times c}$ the target matrix (e.g., one–hot labels for c classes). An RdNN draws a single hidden layer of h units by sampling a *frozen* weight matrix $W \sim \mathcal{D}_W \subset \mathbb{R}^{d\times h}, b \in \mathbb{R}^h$, from a prescribed base distribution (classically i.i.d. uniform or Gaussian, independent across coordinates), and then *never updates* these parameters. With an elementwise nonlinearity $\phi : \mathbb{R} \to \mathbb{R}$, the hidden representation is

$$H \;=\; \phi\big(XW + \mathbf{1}_m b^\top\big) \;\in\; \mathbb{R}^{m\times h}, \tag{8.1}$$

where $\mathbf{1}_m$ denotes an m–vector of ones broadcasting the bias. Training reduces to fitting only the output weights $\Theta \in \mathbb{R}^{h\times c}$ by ridge regression:

$$\Theta^\star \;\in\; \arg\min_{\Theta\in\mathbb{R}^{h\times c}} \|H\Theta - Y\|_F^2 + \frac{\lambda}{2}\|\Theta\|_F^2, \qquad \lambda \geq 0, \tag{8.2}$$

which admits the standard closed forms

$$\text{(primal)} \quad \Theta^\star = (H^\top H + \lambda I_h)^{-1} H^\top Y, \tag{8.3}$$

$$\text{(dual)} \quad \Theta^\star = H^\top (H H^\top + \lambda I_m)^{-1} Y. \tag{8.4}$$

Thus *no backpropagation is required*: once W and b are sampled, training is a single linear solve. This setting covers standard random vector functional link (RVFL), broad learning system (BLS), and deep RVFL (dRVFL) models, all of which use frozen randomized representations followed by efficient output-layer estimation.

8.2.2 Copulas

Copulas are a classical device for isolating and modeling *dependence* separately from *marginals* [22, 58]. Sklar's seminal result shows that any multivariate distribution can be written as a coupling of its univariate marginals through a single function on the unit hypercube, thereby reducing multivariate modeling to estimating marginals and a dependence map on $[0,1]^d$ [21].

8.2.2.1 Copula

Let $\mathbf{X} = (X_1, \ldots, X_d)$ be a random vector with marginal cumulative distribution functions (CDFs) $F_j(t) = \Pr[X_j \le t]$ for $j = 1, \ldots, d$. The *copula associated with* $\mathbf{X}$ is the joint CDF of the transformed vector $(F_1(X_1), \ldots, F_d(X_d)) \in [0,1]^d$, namely

$$\begin{gathered} C(u_1, \ldots, u_d) = \Pr\big\{F_1(X_1) \le u_1, \ldots, F_d(X_d) \le u_d\big\}, \\ (u_1, \ldots, u_d) \in [0,1]^d. \end{gathered} \tag{8.5}$$

Equivalently, a d–variate copula is a distribution function on $[0,1]^d$ whose univariate marginals are uniform on $(0,1)$. Figure 8.1 provides an intuitive picture of the concept.

8.2.2.2 Invariance

If ψ_j are strictly increasing for each j, then $(\psi_1(X_1), \ldots, \psi_d(X_d))$ has the same copula as $(X_1, \ldots, X_d)$. Thus a copula depends only on cross-coordinate order, not on units or scales.

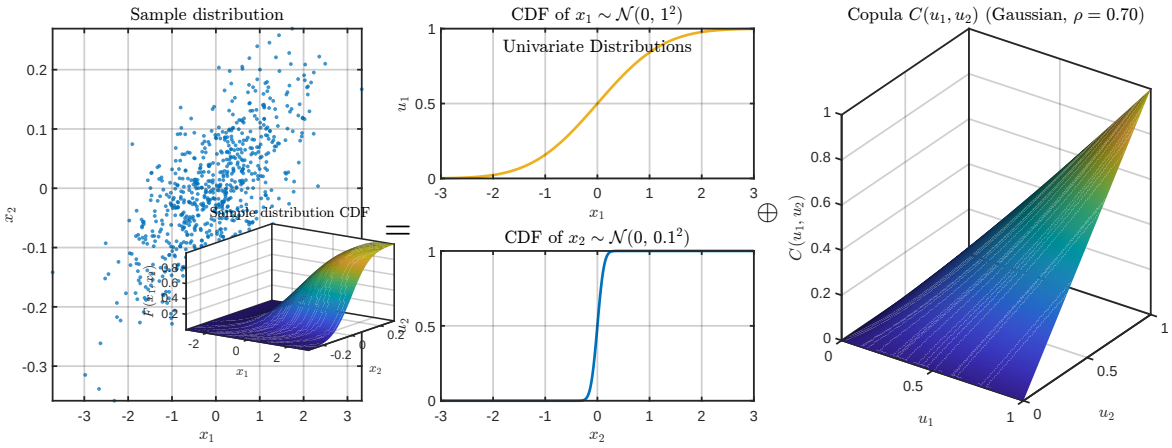


Figure 8.1: Schematic of a copula. *Left:* empirical sample (with its joint CDF inset). *Middle:* univariate CDFs F_1 and F_2 (mapping $x_j \mapsto u_j = F_j(x_j)$). *Right:* the copula surface $C(u_1, u_2)$, i.e., the joint CDF on $[0, 1]^2$ with uniform marginals. The equality sign indicates Sklar's factorization $F(x_1, x_2) = C(F_1(x_1), F_2(x_2))$; the $\oplus$ highlights that C combines the marginal information into a dependence structure.

8.2.2.3 Sklar's Theorem

Let $F : \mathbb{R}^d \to [0, 1]$ be the joint CDF of $\mathbf{X} = (X_1, \ldots, X_d)$ with marginals $F_1, \ldots, F_d$. There exists a copula C such that, for all $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$,

$$F(x_1, \ldots, x_d) \;=\; C\big(F_1(x_1), \ldots, F_d(x_d)\big).$$

If each F_j is continuous, the copula C is unique. Conversely, for any copula C and any marginals $F_1, \ldots, F_d$, the mapping above defines a valid d-variate distribution. If $X_1, \ldots, X_d$ are mutually independent, the copula is the *product copula* $C(u_1, \ldots, u_d) = \prod_{j=1}^{d} u_j$, representing absence of dependence among coordinates.

8.3 Problem Statement

Randomized neural networks (RdNNs) *sample once and freeze* the input-to-hidden weight matrix $W \in \mathbb{R}^{d \times h}$ and bias $b \in \mathbb{R}^h$. With $H = \phi(XW + \mathbf{1}_m b^\top)$ and $\Theta^\star(W) = \arg\min_\Theta \{\|H\Theta - Y\|_F^2 + \lambda\|\Theta\|_F^2\}$, training reduces to a single ridge-regularized linear

solve—no backpropagation is required.

8.3.1 Limitation

The longstanding bottleneck is how W is drawn: almost universally from an *i.i.d.*, factorized base law (uniform or Gaussian). Equivalently, this imposes the *product copula* $C_\perp(u) = \prod_{j=1}^{d} u_j$ across weight coordinates, irrespective of the empirical dependence in the inputs X (correlations, asymmetries, tail dependence). The resulting mismatch can (i) reduce the diversity/effective rank of H, (ii) fail to excite discriminative co-movements, and (iii) worsen the conditioning of $H^\top H + \lambda I$.

Goal. Replace the independence assumption on frozen weights *without* altering the RdNN pipeline. Given data $X \in \mathbb{R}^{m\times d}$, width h, activation ϕ, and ridge λ, training remains the same closed form; only the *sampling law* for W changes. We seek a *one-shot, data-aware* sampler $\mathcal{S}$ that maps X to a distribution over W such that each column $w \in \mathbb{R}^d$ has a joint law whose *copula* matches an estimator of input dependence, while its marginals are simple and well-behaved:

$$\begin{gathered}\mathrm{Copula}(w) \approx \widehat{\mathrm{Copula}}(X), w_j \sim G \text{ for a fixed marginal } G \\ (\text{e.g., } \mathcal{N}(0,1) \text{ or } \mathcal{U}[-1,1]),\ j = 1,\dots,d.\end{gathered} \tag{8.6}$$

Biases b may be drawn independently from a simple one-dimensional law.

Design objective. Among samplers satisfying the dependence-matching and marginal constraints, choose $\mathcal{S}$ to improve downstream risk *without* any iterative tuning of W:

$$\begin{aligned}&\min_{\mathcal{S}} \mathbb{E}_{W\sim\mathcal{S}(X)}\, \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\,\ell\big(h(x;W,b)\,\Theta^\star(W),\, y\big)\big] \\ &\text{s.t. } \mathrm{Copula}(w) \approx \widehat{\mathrm{Copula}}(X), \quad w_j \sim G,\ \ j = 1,\dots,d,\end{aligned} \tag{8.7}$$

where $h(x;W,b) = \phi(xW + b) \in \mathbb{R}^h$ and $\mathcal{D}$ is the data distribution. In short, the problem

is to replace i.i.d. frozen weights with *dependence-aware* frozen weights that align with the empirical copula of X, thereby addressing the core weakness of random initialization while preserving the hallmark efficiency of RdNNs.

8.4 Method

We introduce *Copula-Aligned Weight Initialization* (CAWI), a data-aware procedure that samples the frozen hidden weights in randomized neural networks using only the inputs X. CAWI preserves the classical RdNN pipeline (no backpropagation; closed-form output layer) while aligning the *dependence* among weight coordinates with that of the inputs. Concretely, CAWI (i) maps each feature to $[0, 1]$ via its empirical cumulative distribution function to remove marginal scales, (ii) fits a multivariate copula on $[0, 1]^d$ to summarize input dependence, and (iii) samples each weight column by drawing u from the fitted copula and transforming coordinates through a fixed marginal law G (e.g., $\mathcal{N}(0, 1)$ or $\mathcal{U}[-1, 1]$) via inverse CDFs. The training objective and closed-form solver remain unchanged; only the law used to sample (and then freeze) W is replaced. The detailed CAWI procedure and its steps are presented next. Algorithm 8.1 summarizes the procedure.

8.4.1 Step 1: Probability-Integral Transform of Features

Given $X \in \mathbb{R}^{m\times d}$ with columns $X_{:j}$, we construct pseudo-observations $U_{ij} = \widehat{F}_j(X_{ij}) \in (0, 1)$ with $U \in (0, 1)^{m\times d}$, where $\widehat{F}_j$ is the empirical cumulative distribution function (ECDF) of feature j, computed across samples (i.e., with j fixed and $i = 1, \ldots, m$). A common rank-based implementation sets $U_{ij} = \frac{\mathrm{rank}(X_{ij})}{m+1}$, where ranks are assigned feature-wise across samples. This guarantees $U_{ij} \in (0, 1)$ (never exactly 0 or 1) and thus avoids infinities when applying inverse CDFs in Step 3.

This transformation removes units and strictly monotone re-scalings, isolating cross-

feature *dependence* in the joint distribution of U, consistent with standard empirical copula construction.

8.4.2 Step 2: Fit a Multivariate Copula on U

Let $\mathcal{C} = \{ C(\cdot;\eta) : \eta \in \Xi \}$ be a parametric family of d-variate copulas on $[0,1]^d$, with parameter η in parameter space Ξ. Using the pseudo-observations U, we estimate $\widehat{\eta}$ to obtain a fitted copula $\widehat{C}(\cdot) = C(\cdot;\widehat{\eta})$, which summarizes the empirical dependence of the inputs and has uniform marginals by construction. Estimation is rank-based: one may maximize the copula pseudo-likelihood, solve method-of-moments equations (e.g., matching Kendall's τ), or employ composite likelihoods over pairs when d is large. This fit is performed *once* per training split and is agnostic to the downstream task. The procedure is copula-agnostic: any d-variate copula family for which an estimator and a sampler are available (elliptical or Archimedean) can be plugged into $\mathcal{C}$. Concrete instantiations used in our work are detailed in 8.4.5.

8.4.3 Step 3: Sample Dependence-Aligned Weight Columns

Choose a continuous one-dimensional marginal law G with a strictly increasing quantile G^{-1} for individual weight entries (e.g., $\mathcal{N}(0,1)$ or $\mathcal{U}[-1,1]$). For each hidden unit $t = 1,\dots,h$:

1. draw $u^{(t)} \sim \widehat{C}$ in $[0,1]^d$;
2. set $w^{(t)} \leftarrow G^{-1}\big(u^{(t)}\big)$ coordinatewise, i.e., $w_j^{(t)} \leftarrow G^{-1}(u_j^{(t)})$ for $j = 1,\dots,d$;
3. assign the tth column of W as $W_{:t} \leftarrow w^{(t)}$ and draw a scalar bias b_t independently from a simple one-dimensional law.

Since G^{-1} is strictly increasing for continuous G, the *joint law of the coordinates of* $w^{(t)}$ has copula $\widehat{C}$ while its univariate marginals are G, thereby matching input dependence while retaining convenient weight marginals.

8.4.4 Step 4: Closed-Form Training (Unchanged)

With the dependence-aligned weights W, we form the hidden matrix $H = \phi\big(XW + \mathbf{1}_m b^\top\big)$ and obtain the output layer by the standard ridge-regularized least squares objective $\Theta^\star = \arg\min_\Theta \|H\Theta - Y\|_F^2 + \frac{\lambda}{2}\|\Theta\|_F^2$. No iterative updates of W are introduced: CAWI modifies only the sampling law used to freeze the hidden weights; the training and inference pipeline remains identical to a classical RdNN.

Algorithm 8.1 CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks

Require: Data $X \in \mathbb{R}^{m\times d}$, width h, activation ϕ, ridge λ, weight marginal G, copula family $\mathcal{C}$.
Ensure: Frozen weights $W \in \mathbb{R}^{d\times h}$, bias vector $b \in \mathbb{R}^h$, output weights $\Theta^\star$.
1: Compute pseudo-observations $U_{ij} \leftarrow \widehat{F}_j(X_{ij})$ for every feature j and sample i.
2: Estimate copula parameter $\widehat{\eta}$ from U and set $\widehat{C}(\cdot) \leftarrow C(\cdot;\widehat{\eta})$.
3: **for** $t = 1$ to h **do**
4: Draw $u^{(t)} \sim \widehat{C}$ in $[0,1]^d$.
5: **for** $j = 1$ to d **do**
6: Set $w_j^{(t)} \leftarrow G^{-1}(u_j^{(t)})$.
7: **end for**
8: Set $W_{:t} \leftarrow w^{(t)}$ and sample bias b_t independently.
9: **end for**
10: Form hidden features $H \leftarrow \phi(XW + \mathbf{1}_m b^\top)$.
11: Compute $\Theta^\star$ by ridge-regularized least squares.
12: **return** W, b, and $\Theta^\star$.

8.4.5 Copula Families: Parameterization and Estimation

We instantiate Step 2 with five widely used d-variate copula families that together capture symmetric vs. asymmetric dependence and both light- and heavy-tailed co-movement:

Gaussian (elliptical, tail-neutral), t (elliptical, heavy-tailed), *Clayton* (Archimedean, lower-tail dependent), *Gumbel* (Archimedean, upper-tail dependent), and *Frank* (Archimedean, tail-neutral, non-elliptical). This selection balances expressive coverage with tractable rank-based estimation and efficient sampling, enabling CAWI to adapt to diverse empirical dependence while keeping the training pipeline simple and stable.

8.4.5.1 Elliptical Families (Gaussian, t)

Both families are parameterized by a correlation matrix $R \in \mathbb{R}^{d\times d}$; the t copula additionally has degrees of freedom $\nu > 2$. We estimate R from pairwise Kendall's τ computed on U via the elliptical identity

$$\tau_{ij} \;=\; \tfrac{2}{\pi}\arcsin(R_{ij}) \Rightarrow \widehat{R}_{ij} \;=\; \sin\big(\tfrac{\pi}{2}\,\widehat{\tau}_{ij}\big)\text{(elementwise)},$$

followed by a nearest-correlation projection to enforce symmetry, unit diagonal, and positive (semi)definiteness (a vanishing ridge may be added if needed for numerical stability). For the t copula, we then profile a rank-based pseudo-likelihood over ν with R fixed to obtain $\widehat{\nu}$ (a one-dimensional search). The fitted copula $\widehat{C}$ yields draws $U \in (0,1)^{N\times d}$, which are mapped to weights through the inverse marginals in Step 3.

8.4.5.2 Archimedean Families (Clayton, Gumbel, Frank)

These families are governed by a single scalar parameter θ that controls concordance and tail behavior. We employ robust, composite, rank-based estimation from pairwise Kendall's τ computed on U. Let $\widehat{\tau}_{ij}$ denote the sample Kendall's τ between coordinates i and j, and let $\bar{\tau}$ be their average over $i < j$. With this setup, we estimate a single θ per family as follows.

- **Clayton** ($\theta > 0$, lower-tail dependent). The τ–θ relation $\tau = \theta/(\theta+2)$ yields

$\widehat{\theta} = 2\bar{\tau}/(1-\bar{\tau})$, or, equivalently, one may minimize a composite method-of-moments objective $\sum_{i<j}\left(\frac{\theta}{\theta+2} - \widehat{\tau}_{ij}\right)^2$.

- **Gumbel** ($\theta \geq 1$, upper-tail dependent). With $\tau = 1 - \frac{1}{\theta}$, $\widehat{\theta} = 1/(1-\bar{\tau})$, or via the same composite fit $\sum_{i<j}\left(1 - \frac{1}{\theta} - \widehat{\tau}_{ij}\right)^2$.

- **Frank** ($\theta \in \mathbb{R} \setminus \{0\}$, tail-neutral, non-elliptical). Kendall's τ satisfies $\tau(\theta) = 1 - \frac{4}{\theta} + \frac{4}{\theta}D_1(\theta)$, where D_1 is the Debye function of order 1. We obtain $\widehat{\theta}$ by numerically inverting $\tau(\theta)$ at $\bar{\tau}$ (monotone one-dimensional root-finding), or by composite pseudo-likelihood.

For $d > 2$ we adopt the standard exchangeable extension, i.e., a single global parameter $\widehat{\theta}$ shared across dimensions. This delivers stable, scalable d-variate fits and admits efficient generator-based sampling of $U \in (0,1)^{N\times d}$. As in Step 3, the sampled U is then mapped coordinatewise through the chosen marginal quantile G^{-1} to produce dependence-aligned weight columns.

8.4.6 Computational Complexity of CAWI

CAWI adds a one-time preprocessing stage (Steps 1-3) on top of the RdNN pipeline. Let m denote the number of samples, d the input dimensionality, and h the width (number of hidden units).

8.4.6.1 Step 1: Probability-Integral Transform

Computing the empirical CDF or rank transform for each of the d feature columns requires sorting m values per column, leading to a total cost of $O(d\, m \log m)$.

8.4.6.2 Step 2: Copula Fitting

Using rank-based Kendall's τ for dependence estimation requires computing pairwise concordance statistics for all $d(d-1)/2 = O(d^2)$ feature pairs. A standard implementation of Kendall's τ costs $O(m^2)$ per pair, yielding an overall complexity of $O(d^2m^2)$. For elliptical copulas (Gaussian or t), an additional nearest-correlation or spectral projection step contributes $O(d^3)$. The t-copula further requires a one-dimensional search over degrees of freedom, with per-iteration cost $O(md^2)$, which remains negligible relative to the dominant $O(d^2m^2)$ term. For Archimedean copulas (Clayton, Gumbel, Frank), estimating a single scalar parameter from averaged Kendall's $\bar{\tau}$ incurs negligible cost beyond the $O(d^2m^2)$ computation of the pairwise τ-values.

8.4.6.3 Step 3: Sampling Dependence-Aligned Weight Columns

For elliptical copulas, sampling requires an initial factorization of the $d \times d$ correlation matrix, costing $O(d^3)$, followed by h matrix-vector multiplications, each costing $O(d^2)$, for a total of $O(d^3 + hd^2)$. For Archimedean copulas, sampling each column requires only $O(d)$ operations, yielding a total cost of $O(hd)$. In both cases, mapping the copula samples through the marginal quantile G^{-1} costs $O(hd)$, which is dominated by the previous terms.

8.4.6.4 Step 4: Closed-Form RdNN Training (Unchanged)

With dependence-aligned frozen weights, hidden features are formed as $H = \phi(XW + \mathbf{1}_m b^\top)$. The cost of forming XW is $O(mdh)$, and computing the ridge-regularized output layer using the normal equations costs $O(mh^2 + h^3)$. This is identical to classical RdNNs.

8.4.6.5 Summary

CAWI introduces a one-time preprocessing cost of $O\big(d\, m \log m + d^2m^2 + d^3 + hd^2\big)$, while the dominant runtime in practice remains the standard RdNN training step, $O(mdh + mh^2 + h^3)$. Thus, CAWI preserves the hallmark efficiency of RdNNs. The empirical training-time analysis in Section 8.9 further confirms this efficiency.

8.5 Empirical Results

We evaluate CAWI by integrating it into representative randomized architectures spanning *shallow*, *deep*, and *wide* settings: *(i) RVFL* (single hidden layer with direct links) [16], *(ii) dRVFL* (deep RVFL) [20], and *(iii) BLS* (Broad Learning System; a wide, block-structured random-feature architecture) [19]. We compare the standard i.i.d. baseline against CAWI variants that differ only in the fitted copula used to sample columns of W:

- **i.i.d. baseline**: W is sampled independently from a Uniform distribution ($\mathcal{U}[-1, 1]$).
- **CAWI (elliptical)**: W is sampled using *Gaussian* and t copulas.
- **CAWI (Archimedean)**: W is sampled using *Clayton*, *Frank*, and *Gumbel* copulas.

The detailed experimental setup is provided in Section 8.6.

Table 8.1: Results on the BreaKHis dataset for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	71.0138	**73.2616**	72.3041	69.7440	70.0666	71.0292	**+2.2478**
AD vs. LC	62.1429	62.9507	**65.0425**	62.9932	63.7500	63.4099	**+2.8996**
AD vs. MC	63.6364	65.0909	**66.5455**	64.3636	65.0909	65.4545	**+2.9091**
AD vs. PC	60.2211	61.4796	62.2959	61.4371	62.3044	**62.7126**	**+2.4915**
FA vs. DC	63.5955	**65.3933**	**65.3933**	**65.3933**	64.2697	64.2697	**+1.7978**
FA vs. LC	65.7766	66.3027	**67.3622**	66.0324	66.0396	66.8432	**+1.5856**
FA vs. MC	59.8464	59.6025	60.3403	60.8461	**61.0720**	60.5932	**+1.2256**
FA vs. PC	68.0000	67.7333	**69.0667**	66.6667	67.7333	67.4667	**+1.0667**
PT vs. DC	67.1731	70.2933	71.1971	**71.8125**	69.9760	70.2933	**+4.6394**
PT vs. LC	57.0980	**62.2980**	60.2824	59.0510	59.0980	60.2667	**+5.2000**
PT vs. MC	61.9987	62.3120	61.9799	61.6228	**63.7343**	60.9398	**+1.7356**
PT vs. PC	57.3569	58.1098	59.6941	58.5098	**59.7176**	58.1098	**+2.3607**
TA vs. DC	65.3995	65.9877	66.5847	**67.1817**	66.8745	66.8832	**+1.7822**
TA vs. LC	62.1663	62.8931	**63.6897**	63.6338	62.5577	62.5646	**+1.5234**
TA vs. MC	57.1751	59.5311	59.8418	58.5254	**60.8701**	58.5537	**+3.6950**
TA vs. PC	56.3382	56.7296	58.5674	57.4284	59.7065	**61.9427**	**+5.6045**

Table 8.2: Results on the BreaKHis dataset for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	74.1987	71.0087	69.7542	66.2366	**75.2514**	66.2366	**+1.0527**
AD vs. LC	62.5595	61.7602	61.3350	**67.5170**	65.8503	61.3180	**+4.9575**
AD vs. MC	67.2727	63.6364	65.0909	67.7273	**68.9091**	67.0909	**+1.6364**
AD vs. PC	65.9949	56.5391	56.5391	56.9473	**68.9541**	56.9473	**+2.9592**
FA vs. DC	66.5169	64.7191	66.0674	**67.6338**	67.5577	67.5646	**+1.1169**
FA vs. LC	69.7802	67.9063	67.6505	68.1802	68.4505	**70.0541**	**+0.2739**
FA vs. MC	58.3740	58.3740	59.1117	58.3740	58.3740	**60.5781**	**+2.2041**
FA vs. PC	72.8000	70.2933	73.1971	**74.8125**	72.9760	73.2933	**+2.0125**
PT vs. DC	64.4135	72.4375	68.7260	64.4135	**77.3846**	64.4135	**+12.9711**
PT vs. LC	55.8980	57.4667	56.3137	**57.8824**	57.0980	56.2745	**+1.9844**
PT vs. MC	61.9862	60.2130	60.5576	**65.8521**	61.2657	60.5576	**+3.8659**
PT vs. PC	54.9725	54.5725	**57.7412**	55.3804	54.9725	56.1725	**+2.7687**
TA vs. DC	61.5496	61.5496	61.5496	**62.2137**	61.5496	61.5496	**+0.6641**
TA vs. LC	68.1831	63.6408	63.7107	**69.2802**	67.0510	65.9329	**+1.0971**
TA vs. MC	57.1751	56.8418	56.8418	**60.8531**	56.8418	56.8418	**+3.6780**
TA vs. PC	57.0860	58.6164	**61.1810**	57.8407	57.4563	57.0720	**+4.0950**

Table 8.3: Results on the Schizophrenia ROI datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	60.3448	61.7701	63.0805	63.1034	62.4138	**63.1264**	**+2.7816**
SZ–ROI GM	62.4138	**63.7931**	62.4368	62.4598	61.7931	61.1264	**+1.3793**
SZ–ROI WM	61.0805	60.3448	61.0575	**62.3908**	61.0345	59.6782	**+1.3103**

Table 8.4: Results on the Schizophrenia ROI datasets for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	71.2874	**74.7816**	72.6437	71.9770	71.9310	72.6437	**+3.4942**
SZ–ROI GM	68.5287	69.3103	70.5747	68.5287	**70.6897**	68.5747	**+2.1610**
SZ–ROI WM	63.7701	64.4828	67.1494	**67.2184**	65.8391	66.5287	**+3.4483**

8.5.1 Dataset Description

The evaluation is conducted on 83 benchmark datasets downloaded from the UCI repository [80], comprising 30 binary and 53 multiclass classification tasks. The number of classes ranges from 2 to 26. Sample sizes span from 24 to 58,000 instances, and the number of features ranges from 3 to 263, covering both small- and large-scale datasets as well as low- and high-dimensional regimes. Detailed dataset statistics are provided in Table 8.6. To validate performance in real-world settings, we additionally evaluate CAWI on two biomedical datasets: the *BreaKHis* breast cancer dataset [83] and *Schizophrenia ROI* dataset from the COBRE repository (`http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html`). For BreaKHis, we use 1,240 histopathological image scans at $400\times$ magnification, organized into benign and malignant classes. The benign subclasses are adenosis (AD, 106), fibroadenoma (FA, 237), phyllodes tumor (PT, 115), and tubular adenoma (TA, 130); the malignant subclasses are lobular carcinoma (LC, 137), papillary carcinoma (PC, 138), ductal carcinoma (DC, 208), and mucinous carcinoma (MC, 169). Feature extraction follows the procedure in [84]. The Schizophrenia ROI dataset includes 72 schizophrenia subjects (ages 18–65; mean 38.1 ± 13.9 years) and 74 healthy controls (ages 18–65; mean 35.8 ± 11.5 years); ROI features are derived following [158].

8.5.2 Performance Evaluation

In this subsection, we present empirical results on three groups of benchmarks: the BreaKHis dataset, the Schizophrenia ROI dataset, and a broader suite of 83 UCI benchmark datasets. For BreaKHis and Schizophrenia ROI, we evaluate RVFL, dRVFL, and BLS under the standard i.i.d. initialization and the proposed CAWI variants (Gaussian, t, Clayton, Frank, and Gumbel). Tables 8.1 and 8.2 report the BreaKHis results for RVFL and dRVFL, while Tables 8.3 and 8.4 present the corresponding results for the Schizophrenia ROI dataset. Results for BLS are provided in Tables 8.7–8.8. For the broader benchmark evaluation, we report results on 83 UCI datasets (30 binary and 53 multiclass). The aggregate summary is presented in Table 8.5, while the complete per-dataset results are reported in Tables 8.9–8.10. All these results are reported using a fixed seed (`rng(42, 'twister')`); additional experiments with two independent seeds (7 and 123) are provided in Section 8.7 to assess seed sensitivity.

8.5.2.1 Results on BreaKHis Dataset

Table 8.1 shows that *every* pairwise task attains a higher test accuracy with a copula-aligned initialization than with the i.i.d. baseline (rightmost column > 0 for all 16 rows). Gains range from about $+1.07$ to $+5.60$ percentage points (e.g., TA vs. PC: $+5.60$, PT vs. LC: $+5.20$, PT vs. DC: $+4.64$), indicating that dependence-aware sampling of W provides a meaningful lift even in a shallow RVFL. Not every copula family dominates on every task; rather, the *best* variant shifts across pairs, consistent with heterogeneous dependence patterns in the features. Across the 16 pairs, the winning family varies: the t-copula is best on 5 tasks (AD vs. LC/MC; FA vs. LC; TA vs. LC; plus a three-way tie in FA vs. DC), *Gaussian* on 3 (e.g., AD vs. DC; FA vs. PC; PT vs. LC), *Clayton* on 2 (PT vs. DC; TA vs. DC), *Frank* on 4 (FA vs. MC; PT vs. MC/PC), and *Gumbel* on 2 (AD vs. PC; TA vs. PC). A three-way

tie (Gaussian/t/Clayton) in FA vs. DC suggests that distinct copula families can approximate the relevant dependence equally well in some cases. Overall, every BreaKHis pair benefits from copula-aligned initialization (all improvements > 0), with gains up to $+5.60$ percentage points, reinforcing the central claim: injecting empirical dependence into the frozen projections consistently improves RVFL performance, while different copula families provide complementary advantages across tasks. Table 8.2 shows that copula-aligned initialization again improves over the i.i.d. baseline on *every* pair (rightmost column > 0 for all 16 rows), now with a wider spread of gains: from $+0.27$ to $+12.97$ percentage (e.g., PT vs. DC: $+12.97$, TA vs. PC: $+4.10$, PT vs. MC: $+3.87$). This indicates that dependence-aware sampling remains beneficial in deeper randomized stacks and can yield larger lifts when multiple frozen layers compound representational biases. The winning family varies by pair, with a clear pattern favoring Archimedean copulas: *Clayton* leads on **8** tasks, *Frank* on **4**, *Gumbel* on **2**, and the t-copula on **2**; *Gaussian* does not take a win in this setting. These outcomes are consistent with heterogeneous, often asymmetric dependence in the extracted features: Clayton/Frank/Gumbel (which capture different tail/asymmetry profiles) dominate, while elliptical models win when dependence is more symmetric.

8.5.2.2 Results on Schizophrenia Dataset

Across the three ROI configurations (GM+WM, GM, WM), CAWI improves upon the i.i.d. baseline in both architectures. In the RVFL setting (see Table 8.3), the best copula varies by ROI: *Gumbel* leads on GM+WM ($+2.78$), *Gaussian* on GM ($+1.38$), and *Clayton* on WM ($+1.31$). For dRVFL (see Table 8.4), the pattern shifts: *Gaussian* tops GM+WM ($+3.49$ pp), *Frank* on GM ($+2.16$), and *Clayton* again wins on WM ($+3.45$). Overall, these results echo the BreaKHis findings without redundancy: dependence-aware sampling of W consistently lifts accuracy, and the most effective copula family depends on the ROI subset

and architecture–underscoring the value of modeling the *type* of dependence (symmetric vs. asymmetric; central vs. tail) rather than assuming independence.

8.5.3 Results on 83 UCI Datasets

On the broader benchmark suite, we evaluate RVFL on 83 UCI datasets. To provide a concise and transparent summary of the empirical findings, we report the aggregate performance of CAWI in Table 8.5. For each dataset, we compare the conventional i.i.d. initialization with the best-performing CAWI configuration (i.e., the copula family yielding the highest test accuracy for that dataset). This reflects the intended usage of CAWI as a flexible, copula-agnostic framework capable of adapting to dataset-specific dependency structures. Across all 83 datasets, the best CAWI variant outperforms the i.i.d. baseline on every dataset. The average test accuracy of the i.i.d. initialization is 78.08%, whereas the average accuracy of the best CAWI configuration reaches 79.35%, corresponding to a mean improvement of $+1.27$ percentage points. These results demonstrate that CAWI yields consistent improvements across both binary and multiclass problems, rather than isolated gains on a small subset of datasets.

Table 8.5: Aggregate performance summary across 83 UCI datasets.

Metric	**Value**
Mean accuracy (i.i.d.)	78.08%
Mean accuracy (Best CAWI)	79.35%
Mean improvement	+1.27%
Datasets where CAWI $>$ i.i.d.	83 / 83

8.6 Experimental Setup and Hyperparameter Setting

All the experiments are implemented using MATLAB R2023a and executed on a Windows 10 PC equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz (4 cores, 8

logical processors) and 16 GB RAM. Each dataset is preprocessed by normalizing the input features to have zero mean and unit variance. The description of 83 datasets (binary and multiclass) used in the experiments are provided in Table 8.6. A 5-fold cross-validation procedure is employed to ensure reliable and unbiased evaluation. In each fold, the dataset is split into 80% training data and 20% testing data. For every combination of hyperparameters, the model is trained on the training data and evaluated on the testing data across all 5 folds. The testing accuracy is recorded for each fold. The final testing accuracy for each dataset is computed as the mean testing accuracy across the five folds, providing a robust estimate of the model's performance.

To eliminate any possibility of data leakage, all copula estimation steps are performed strictly within each training split. For every fold, the pseudo-observations and copula parameters are computed exclusively using the training features of that fold. The fitted copula is then used to sample the hidden-layer weights, which are subsequently evaluated on the disjoint test set. At no stage is information from the test portion used during copula fitting or weight initialization.

Hyperparameter tuning is performed using a grid search strategy to identify the settings for each model. For each model, the regularization parameter (λ) is selected from $\{10^i \mid i = -6, -5, \ldots, 6\}$. For RVFL, the number of hidden nodes (h) varies from $[3 : 20 : 203]$, and seven activation functions (Sigmoid (1), Sine (2), Tribas (3), Radbas (4), Tansig (5), ReLU (6), and SELU (7)) are evaluated. For BLS, the number of feature windows (q), the number of feature nodes in each window (p), and the number of enhancement nodes (r) follow the standard BLS experimental setting, with Tansig as the activation function. For dRVFL, we adopt the same hyperparameter settings as provided in [20] and evaluate three activation functions: Sigmoid (1), ReLU (2), and SELU (3).

Table 8.6: Description of binary and multiclass datasets used in the experiments. The table lists the dataset name, number of samples, number of features, and number of classes.

Dataset	Number of Samples	Number of Features	Number of Classes	Dataset	Number of Samples	Number of Features	Number of Classes
Multiclass datasets							
abalone	4177	8	3	synthetic_control	600	60	6
annealing	898	31	5	thyroid	7200	21	3
arrhythmia	452	262	13	vertebral_column_3clases	310	6	3
balance_scale	625	4	3	wall_following	5456	24	4
cardiotocography_10clases	2126	21	10	waveform	5000	21	3
cardiotocography_3clases	2126	21	3	waveform_noise	5000	40	3
conn_bench_vowel_deterding	990	11	11	wine	178	13	3
contrac	1473	9	3	wine_quality_red	1599	11	6
dermatology	366	34	6	wine_quality_white	4898	11	7
ecoli	336	7	8	yeast	1484	8	10
energy_y1	768	8	3	zoo	101	16	7
energy_y2	768	8	3	**Binary datasets**			
flags	194	28	8	bank	4521	16	2
glass	214	9	6	blood	748	4	2
heart_cleveland	303	13	5	breast_cancer	286	9	2
heart_switzerland	123	12	5	breast_cancer_wisc	699	9	2
image_segmentation	2310	18	7	breast_cancer_wisc_diag	569	30	2
iris	150	4	3	breast_cancer_wisc_prog	198	33	2
led_display	1000	7	10	chess_krvkp	3196	36	2
lenses	24	4	3	congressional_voting	435	16	2
letter	20000	16	26	conn_bench_sonar_mines_rocks	208	60	2
low_res_spect	531	100	9	credit_approval	690	15	2
lymphography	148	18	4	cylinder_bands	512	35	2
molec_biol_splice	3190	60	3	echocardiogram	131	10	2
nursery	12960	8	5	haberman_survival	306	3	2
oocytes_merluccius_states_2f	1022	25	3	hepatitis	155	19	2
oocytes_trisopterus_states_5b	912	32	3	horse_colic	368	25	2
optical	5620	62	10	ilpd_indian_liver	583	9	2
page_blocks	5473	10	5	ionosphere	351	33	2
pendigits	10992	16	10	monks_1	556	6	2
pittsburg_bridges_MATERIAL	106	7	3	monks_3	554	6	2
pittsburg_bridges_REL_L	103	7	3	oocytes_merluccius_nucleus_4d	1022	41	2
pittsburg_bridges_SPAN	92	7	3	oocytes_trisopterus_nucleus_2f	912	25	2
pittsburg_bridges_TYPE	105	7	6	pima	768	8	2
post_operative	90	8	3	planning	182	12	2
seeds	210	7	3	spect	265	22	2
semeion	1593	256	10	spectf	267	44	2
soybean	683	35	18	statlog_australian_credit	690	14	2
statlog_image	2310	18	7	statlog_german_credit	1000	24	2
statlog_landsat	6435	36	6	statlog_heart	270	13	2
statlog_shuttle	58000	9	7	tic_tac_toe	958	9	2
statlog_vehicle	846	18	4	titanic	2201	3	2

Table 8.7: Results on the BreaKHis dataset for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.7066	71.0394	70.8966	69.7440	**72.3041**	71.8292	**+1.5975**
AD vs. LC	63.1429	62.9507	**65.0425**	62.9932	63.7500	63.4099	**+1.8996**
AD vs. MC	63.8396	65.8909	**68.5455**	65.3636	66.0909	66.4545	**+4.7059**
AD vs. PC	61.2956	61.8296	62.2959	62.4371	63.3044	**64.2126**	**+2.9170**
FA vs. DC	64.1055	**65.3933**	**65.3933**	**65.3933**	64.2697	64.2697	**+1.2878**
FA vs. LC	67.8991	68.7099	**68.7135**	67.6468	68.1802	68.7099	**+0.8144**
FA vs. MC	60.8371	62.3156	61.3403	**63.8461**	61.0720	60.5932	**+3.0090**
FA vs. PC	68.5746	67.9333	**69.8667**	66.8667	67.9233	67.8667	**+1.2921**
PT vs. DC	68.1731	70.2933	71.1971	**71.8125**	69.9760	70.2933	**+3.6394**
PT vs. LC	57.5980	**62.2980**	60.2824	59.0510	59.0980	60.2667	**+4.7000**
PT vs. MC	62.5987	62.3120	61.9799	61.6228	**64.5243**	60.9398	**+1.9256**
PT vs. PC	58.1069	58.1098	59.6941	58.5098	**60.0176**	58.1098	**+1.9107**
TA vs. DC	62.9939	64.7937	**66.2818**	65.1141	65.4214	65.7024	**+3.2879**
TA vs. LC	65.9189	66.7086	68.1971	67.3934	67.4074	**68.9099**	**+2.9910**
TA vs. MC	60.1921	61.8814	**63.5593**	63.2260	**63.5593**	61.5254	**+3.3672**
TA vs. PC	61.1740	60.8036	**62.2851**	60.4472	60.0978	61.5933	**+1.1111**

Table 8.8: Results on the Schizophrenia ROI datasets for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	71.9770	72.6897	74.0000	74.0690	74.0230	**74.7126**	**+2.7356**
SZ–ROI GM	70.6667	**73.2414**	71.9540	71.9080	69.9540	72.6207	**+2.5747**
SZ–ROI WM	67.9080	67.2184	**69.2184**	67.9080	67.9080	68.5287	**+1.3104**

Table 8.9: Results on benchmark binary datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
bank	89.6705	89.8032	89.7809	89.6705	**89.8254**	89.7146	**+0.1549**
blood	76.7714	77.0389	**77.5732**	**77.5732**	77.4398	77.4398	**+0.8018**
breast_cancer	70.1754	70.1754	70.1754	70.1754	70.1754	**70.5263**	**+0.3509**
breast_cancer_wisc	88.5612	88.7040	**89.4193**	88.7040	88.9887	89.2775	**+0.8581**
breast_cancer_wisc_diag	94.7260	94.9014	**95.4308**	95.2554	95.0753	95.2523	**+0.7048**
breast_cancer_wisc_prog	81.8462	82.8462	**83.4231**	82.3718	82.8974	81.8846	**+1.5769**
chess_krvkp	81.4485	82.4172	82.3546	82.4174	**82.8535**	82.1356	**+1.4050**
congressional_voting	63.9080	63.6782	**64.1379**	63.6782	63.6782	63.6782	**+0.2299**
conn_bench_sonar_mines_rocks	61.9861	65.4123	**66.7944**	62.5900	64.4948	64.3786	**+4.8083**
credit_approval	85.2174	**85.7971**	85.6522	**85.7971**	85.6522	85.6522	**+0.5797**
cylinder_bands	69.7392	69.7335	**70.1085**	69.1376	68.9587	69.7373	**+0.3693**
echocardiogram	85.4416	85.4416	**86.2108**	85.4416	85.4416	85.4416	**+0.7692**
haberman_survival	74.1460	73.4902	**74.8017**	74.4738	74.4738	74.1460	**+0.6557**
hepatitis	85.1613	86.4516	86.4516	85.8065	85.8065	**87.0968**	**+1.9355**
horse_colic	85.8830	86.1570	**86.9604**	86.4198	86.4198	86.1570	**+1.0774**
ilpd_indian_liver	71.7006	72.7306	73.0725	72.3902	**73.5883**	72.7336	**+1.8877**
ionosphere	89.7626	90.0443	**91.1871**	90.0523	90.6117	90.6237	**+1.4245**
monks_1	82.5338	84.8745	**84.8858**	84.1506	84.5206	83.9704	**+2.3520**
monks_3	91.3415	92.0606	92.6011	92.2424	92.0573	**92.6044**	**+1.2629**
oocytes_merluccius_nucleus_4d	82.9775	83.1703	**83.2669**	82.8771	82.8780	83.0693	**+0.2894**
oocytes_trisopterus_nucleus_2f	79.1647	80.5861	80.2600	**81.1355**	80.0342	80.1417	**+1.9708**
pima	73.7043	73.9649	**74.2212**	73.9649	73.1856	73.7051	**+0.5169**
planning	71.3814	71.9369	71.9369	**72.4625**	71.9520	71.9369	**+1.0811**
spect	67.1698	68.3019	**69.8113**	69.0566	68.3019	68.6792	**+2.6415**
spectf	79.3431	79.3501	79.7205	79.7135	**79.7275**	79.3431	**+0.3844**
statlog_australian_credit	68.5507	68.5507	68.9855	**69.2754**	68.9855	68.9855	**+0.7247**
statlog_german_credit	76.8000	77.7000	77.7000	**78.0000**	77.5000	77.6000	**+1.2000**
statlog_heart	80.7407	81.4815	**82.9630**	81.8519	82.5926	81.1111	**+2.2223**
tic_tac_toe	89.9700	90.8077	**91.1229**	90.5923	89.9700	90.3878	**+1.1529**
titanic	77.9168	77.9168	**78.1901**	77.9168	77.9168	77.9168	**+0.2733**

Table 8.10: Results on benchmark multiclass datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold**; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
abalone	63.5141	63.6813	63.8735	63.8728	63.9687	**64.0169**	**+0.5028**
annealing	89.5202	90.3048	**90.4115**	89.8603	90.3048	90.1906	**+0.8913**
arrhythmia	70.8010	71.0256	71.0183	70.8010	**71.2405**	**71.2405**	**+0.4395**
balance_scale	98.5600	98.5600	**98.7200**	**98.7200**	98.5600	**98.7200**	**+0.1600**
cardiotocography_10clases	**70.7455**	69.9461	70.6033	70.3695	70.2276	70.3207	**+0.0000**
cardiotocography_3clases	85.9859	86.3148	**86.6909**	86.5030	86.5023	86.2684	**+0.7050**
conn_bench_vowel_deterding	95.8586	96.1616	96.1616	**96.4646**	96.1616	96.1616	**+0.6060**
contrac	41.0681	**42.7001**	42.0883	41.4034	41.6765	42.0830	**+1.6320**
dermatology	97.2677	97.8119	**98.0859**	97.8119	97.8119	98.0822	**+0.8182**
ecoli	60.6234	61.2160	61.2116	61.2072	61.2160	**61.5057**	**+0.8823**
energy_y1	88.5307	89.1936	**89.9677**	88.6682	89.1945	89.0578	**+1.4370**
energy_y2	90.1002	90.7538	**91.6645**	91.2783	91.0126	91.0160	**+1.5643**
flags	53.6437	55.1822	55.2227	53.6302	**56.2078**	53.6437	**+2.5641**
glass	38.1617	39.0808	40.9413	**42.3367**	39.5570	40.0221	**+4.1750**
heart_cleveland	59.3661	60.0546	**60.7049**	60.3661	60.6940	59.7322	**+1.3388**
heart_switzerland	47.3000	47.9333	49.7000	48.0333	**49.7667**	47.3333	**+2.4667**
image_segmentation	88.1385	88.4848	**88.5714**	88.2251	**88.5714**	88.3117	**+0.4329**
iris	75.3333	**78.6667**	76.6667	76.0000	78.0000	77.3333	**+3.3334**
led_display	72.9000	73.2000	**73.7000**	73.3000	73.6000	73.3000	**+0.8000**
lenses	92.0000	**96.0000**	**96.0000**	**96.0000**	**96.0000**	**96.0000**	**+4.0000**
letter	80.8500	80.9050	80.9750	81.0350	80.9650	**81.1450**	**+0.2950**
low_res_spect	88.5117	88.5117	**89.4534**	89.0778	88.8891	88.8891	**+0.9417**
lymphography	86.4138	**88.5057**	87.7931	87.1034	87.7931	87.8161	**+2.0919**
molec_biol_splice	53.5110	54.9216	**56.3009**	53.8558	53.9812	54.4514	**+2.7899**
nursery	71.2114	70.8951	71.7978	70.9877	71.7747	**72.9784**	**+1.7670**
oocytes_merluccius_states_2f	91.5782	91.9699	**92.3649**	92.0679	91.9699	91.9694	**+0.7867**
oocytes_trisopterus_states_5b	87.9427	89.1491	**89.4806**	89.1509	88.9227	88.9275	**+1.5379**
optical	96.9929	97.2064	**97.2242**	97.1530	97.1886	97.1886	**+0.2313**
page_blocks	95.3771	95.7242	95.8157	95.5597	95.5781	**95.5963**	**+0.4386**
pendigits	98.5353	98.5990	98.6899	98.6626	98.6536	**98.7081**	**+0.1728**
pittsburg_bridges_MATERIAL	78.7013	76.8831	78.6147	76.8831	**80.4762**	78.7013	**+1.7749**
pittsburg_bridges_REL_L	63.1429	63.2381	68.8571	65.2381	66.2381	**66.8095**	**+5.7142**
pittsburg_bridges_SPAN	63.0409	64.2690	**67.4854**	64.2105	64.2105	63.2749	**+4.4445**
pittsburg_bridges_TYPE	42.8571	42.8571	**46.6667**	42.8571	44.7619	42.8571	**+3.8096**
post_operative	71.1111	71.1111	71.1111	71.1111	71.1111	**72.2222**	**+1.1111**
seeds	90.0000	89.5238	**90.4762**	**90.4762**	89.5238	90.0000	**+0.4762**
semeion	87.8208	88.4488	**89.0751**	88.9505	88.6369	88.4484	**+1.2543**
soybean	90.4755	90.1889	**91.0702**	90.1793	90.0333	90.0333	**+0.5947**
statlog_image	95.4113	95.4545	**95.6277**	95.5844	95.5844	95.5844	**+0.2164**
statlog_landsat	82.1134	82.6107	82.5486	82.5175	82.3155	**82.3310**	**+0.4973**
statlog_shuttle	98.7017	98.7448	**98.7534**	98.7190	98.7276	98.7224	**+0.0517**
statlog_vehicle	82.1511	82.5040	82.8597	**82.8611**	82.2694	**82.8611**	**+0.7100**
synthetic_control	54.0000	54.8333	55.1667	55.0000	**55.5000**	55.3333	**+1.5000**
thyroid	95.7639	95.7361	95.9028	95.7917	95.8056	**95.9444**	**+0.1805**
vertebral_column_3clases	65.1613	66.1290	66.4516	**66.7742**	65.8065	66.1290	**+1.6129**
wall_following	77.6955	78.2640	78.1717	78.0253	77.8052	**78.8133**	**+1.1178**
waveform	86.8800	86.9200	87.0000	87.0000	**87.0800**	86.8400	**+0.2000**
waveform_noise	86.3400	86.4200	86.5000	86.4800	86.3800	86.4200	**+0.1600**
wine	96.6349	96.6508	**97.2063**	96.6508	96.6508	96.6508	**+0.5714**
wine_quality_red	59.6638	60.0380	**60.9122**	60.6025	60.6023	60.6019	**+1.2484**
wine_quality_white	52.7576	52.8187	53.0842	52.9617	**53.1046**	52.9615	**+0.3470**
yeast	57.1424	57.7496	57.6169	**57.8840**	57.6815	57.3460	**+0.7416**
zoo	95.0000	**97.0000**	**97.0000**	95.0000	96.0000	96.0000	**+2.0000**

8.7 Multi-Seed Stability Analysis

To assess the robustness of CAWI with respect to randomness, we additionally evaluate the RVFL model under multiple independent random seeds. While all primary experiments were conducted using a fixed seed (`rng(42, 'twister')`), we re-run the experiments using two additional seeds (7 and 123) for both the i.i.d. baseline and all copula-based variants.

The detailed results for the BreaKHis dataset under Seed 7 and Seed 123 are reported in Tables 8.11 and 8.12, respectively. Similarly, the corresponding results for the Schizophrenia ROI dataset are presented in Tables 8.13 and 8.14.

Across both datasets and independent seeds, CAWI consistently outperforms the conventional i.i.d. initialization. Importantly, the relative improvements remain stable in magnitude and direction, indicating that the gains are not attributable to a particular random realization.

Overall, the multi-seed evaluation confirms that CAWI provides stable and reproducible performance improvements across independent random initializations, reinforcing its robustness as a dependency-aware weight initialization strategy.

Table 8.11: BreaKHis dataset (Seed 7): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.6429	72.8934	**73.1562**	69.4815	70.3274	71.2948	**+2.5133**
AD vs. LC	62.4138	**65.2847**	64.7193	63.1562	63.9284	63.7451	**+2.8709**
AD vs. MC	63.9091	65.2727	64.9091	**66.7273**	65.2727	65.6364	**+2.8182**
AD vs. PC	60.5793	61.8926	62.1038	61.6704	**62.9381**	62.4859	**+2.3588**
FA vs. DC	63.2584	65.6742	64.9438	65.1124	**65.8876**	64.6067	**+2.6292**
FA vs. LC	66.1039	66.5783	66.9324	**67.5429**	66.3891	66.2108	**+1.4390**
FA vs. MC	59.6139	60.1847	**61.4893**	60.3284	60.8951	60.2556	**+1.8754**
FA vs. PC	68.1733	67.6267	68.8000	67.2000	**69.4667**	67.9467	**+1.2934**
PT vs. DC	67.5192	70.8365	**72.0479**	71.5841	69.8654	70.4712	**+4.5287**
PT vs. LC	57.2863	61.8745	60.7294	**62.4196**	59.6275	60.5490	**+5.1333**
PT vs. MC	61.7254	**63.8194**	62.4183	61.3562	63.2816	61.4729	**+2.0940**
PT vs. PC	57.0196	58.5882	59.3725	**60.1176**	59.4902	58.6667	**+3.0980**
TA vs. DC	65.7239	66.5104	**67.8529**	66.9274	66.4183	66.2104	**+2.1290**
TA vs. LC	62.0384	63.3827	63.1562	**63.9418**	62.9716	62.3194	**+1.9034**
TA vs. MC	57.4283	59.9274	**61.2638**	58.8951	60.5183	58.1029	**+3.8355**
TA vs. PC	56.5071	57.2849	58.9427	**61.8394**	59.3184	58.4716	**+5.3323**

Table 8.12: BreaKHis dataset (Seed 123): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.3251	**73.1089**	72.5614	69.9123	70.2947	71.3458	**+2.7838**
AD vs. LC	61.8742	63.2184	**64.8936**	62.7451	63.4826	63.1572	**+3.0194**
AD vs. MC	64.1827	65.3636	**66.9091**	64.7273	65.4545	65.0909	**+2.7264**
AD vs. PC	60.8934	61.7238	62.5481	61.1904	62.0917	**63.1549**	**+2.2615**
FA vs. DC	63.9128	**65.7303**	65.1685	65.0562	64.5843	64.4157	**+1.8175**
FA vs. LC	66.2419	66.5783	**67.6194**	66.3891	66.2108	66.5964	**+1.3775**
FA vs. MC	59.4876	59.8512	60.5918	60.6139	**61.3284**	60.8473	**+1.8408**
FA vs. PC	67.6267	67.9733	**69.3200**	66.9067	67.4933	67.7200	**+1.6933**
PT vs. DC	67.8495	70.5614	71.4583	**72.1047**	70.2136	70.5289	**+4.2552**
PT vs. LC	56.7293	**62.5647**	60.5412	59.3176	59.3529	60.4980	**+5.8354**
PT vs. MC	61.5394	62.5893	62.2461	61.8951	**63.4716**	61.1825	**+1.9322**
PT vs. PC	57.7142	58.3725	59.4314	58.2647	**59.9608**	58.3490	**+2.2466**
TA vs. DC	65.0418	66.2349	66.8293	**67.4264**	66.6172	66.5405	**+2.3846**
TA vs. LC	62.5194	63.1508	**63.9271**	63.3915	62.8104	62.8173	**+1.4077**
TA vs. MC	57.5628	59.7834	60.0941	58.7921	**61.1248**	58.8104	**+3.5620**
TA vs. PC	56.7049	56.9863	58.3207	57.6851	59.4632	**61.6894**	**+4.9845**

Table 8.13: Schizophrenia ROI dataset (Seed 7): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	60.7126	61.9540	**63.4253**	62.8506	62.6897	63.0345	**+2.7127**
SZ–ROI GM	61.9540	63.5632	**63.9080**	62.1839	62.0690	61.4023	**+1.9540**
SZ–ROI WM	61.4483	60.5977	**62.5287**	61.8161	61.2931	59.9425	**+1.0804**

Table 8.14: Schizophrenia ROI dataset (Seed 123): Test accuracy (%). Best per row is **bold**; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	59.9080	61.5402	62.8736	**63.3563**	62.1839	62.7586	**+3.4483**
SZ–ROI GM	62.7586	63.4483	62.6437	62.7126	**64.0230**	61.3793	**+1.2644**
SZ–ROI WM	60.8046	**62.6437**	61.2931	61.8621	60.7816	59.4253	**+1.8391**

8.8 Statistical Significance Across Datasets

To assess whether the performance improvements of CAWI over the i.i.d. initialization are statistically significant across datasets, we conduct a non-parametric Wilcoxon signed-rank test over the 83 UCI benchmark datasets.

For each dataset $i \in \{1, \ldots, 83\}$, let A_i^{iid} denote the test accuracy obtained using the conventional i.i.d. initialization, and let A_i^{CAWI} denote the test accuracy achieved by the best-performing CAWI configuration on the same dataset. We use the best CAWI variant per dataset because CAWI is designed as a flexible, copula-agnostic framework: different datasets exhibit different dependency structures, and therefore different copula families may be most appropriate for different tasks. This comparison reflects the intended usage of CAWI, where the copula family is selected to best capture the empirical dependence of each dataset.

We then construct paired observations $\left(A_i^{\text{iid}}, A_i^{\text{CAWI}}\right)$, $i = 1, \ldots, 83$, and apply the Wilcoxon signed-rank test to the differences $D_i = A_i^{\text{CAWI}} - A_i^{\text{iid}}$.

The resulting Wilcoxon statistic is $W = 3403$, while the maximum possible statistic for 83 paired observations is 3486. The corresponding p-value satisfies $p < 10^{-6}$, indicating extremely strong statistical evidence against the null hypothesis of equal median performance.

These results demonstrate that the observed improvements of CAWI over the i.i.d. initialization are not attributable to random fluctuations on individual datasets, but instead reflect a consistent and statistically significant advantage across diverse benchmark problems.

8.9 Empirical Training-Time Analysis

To complement the theoretical complexity discussion provided in Section 8.4.6, we report an empirical evaluation of the training time incurred by CAWI under different copula families. Specifically, we measure the average training time (in seconds) of the RVFL model initialized using five copula families, Gaussian, t, Clayton, Frank, and Gumbel, alongside the conventional i.i.d. initialization baseline.

The results are averaged over all 83 UCI benchmark datasets used in our experimental evaluation. Table 8.15 summarizes the average training time across datasets.

Table 8.15: Average training time (in seconds) across 83 UCI datasets.

Method	i.i.d.	Gaussian	t	Clayton	Frank	Gumbel
Avg. Time (s)	0.00691	0.00824	0.00924	0.00727	0.00733	0.00742

As observed, all copula-based initializations introduce only a marginal computational overhead relative to the i.i.d. baseline. The additional cost stems primarily from the estimation of the empirical copula parameters and subsequent dependency-aligned sampling. However, since these operations scale linearly with the number of features and hidden nodes, the overall training complexity remains effectively unchanged in practice.

Importantly, the empirical overhead remains below 0.003 seconds on average, confirming that CAWI preserves the computational efficiency characteristic of randomized neural networks while improving stability and performance.

8.10 Chapter Summary

In this work, we address a foundational gap in randomized neural networks, namely that input-to-hidden weights are randomly initialized and then held fixed throughout training. We present CAWI, a plug-in initialization scheme that aligns the frozen weight columns

with the empirical dependence structure of the inputs via a fitted copula, while leaving the classical RdNN pipeline–closed-form readout, no backpropagation–unchanged. By modifying only the sampling law for W, CAWI yields dependence-aware hidden projections with a computational footprint comparable to i.i.d. initialization. Building on this design, our empirical study shows consistent and often substantial gains across 83 UCI benchmarks and two biomedical datasets (BreaKHis and Schizophrenia ROI) using RVFL, dRVFL, and BLS architectures, all while preserving backpropagation-free training. Codes are provided at `https://github.com/mtanveer1/CAWI`.

A natural next step for future research is the development of automatic copula selection strategies. Rather than manually evaluating multiple copula families, a data-driven mechanism could identify the most suitable dependency structure for a given dataset and initialize the model accordingly. Such an adaptive selection procedure would further streamline CAWI while preserving its computational efficiency and enhancing its ability to capture dataset-specific dependence patterns.

Chapter 9

Conclusions and Future Directions

9.1 Synthesis of Contributions

This thesis investigated robust and efficient learning under three central sources of difficulty in real-world data: noise, class imbalance, and randomness. These difficulties appear in different forms across learning problems. Noise may enter through erroneous labels, corrupted measurements, outliers, or ambiguous samples. Class imbalance changes the effective contribution of majority and minority classes and may bias the learned decision boundary toward the dominant class. Randomness appears explicitly in randomized neural networks, where a significant part of the representation is generated by randomly assigned and fixed parameters. Although these issues arise from different mechanisms, they have a common effect: they distort the empirical risk minimized during training and can therefore degrade generalization.

The central idea developed in this thesis is that robustness and efficiency can be improved by redesigning the components that directly shape the learning objective. In a broad

regularized empirical-risk form,

$$\min_{f \in \mathcal{F}} \ \Omega(f) + C \sum_{i=1}^{n} s_i \mathcal{L}\left(y_i, f(\mathbf{x}_i)\right), \tag{9.1}$$

the behavior of the model is controlled by the hypothesis class $\mathcal{F}$, the regularizer $\Omega(\cdot)$, the loss function $\mathcal{L}(\cdot, \cdot)$, and the sample weight s_i. Classical learning methods often focus primarily on the model class while keeping the loss function, sample weighting, or random initialization mechanism fixed. In contrast, this thesis shows that substantial improvements can be obtained by carefully designing these components themselves. The thesis develops novel loss functions, an adaptive sample weighting mechanism, and a dependence-aware weight initialization strategy, while also refining optimization strategies to preserve computationally attractive model structures.

The thesis first developed robust loss functions for hyperplane-based learning. The RoBoSS loss introduced boundedness, sparsity, smoothness, and robustness within a supervised classification setting. These properties are important because a bounded loss limits the influence of extreme samples, sparsity encourages learning from informative observations, and smoothness enables efficient gradient-based optimization. This thesis incorporated the RoBoSS loss into support vector machine (SVM) learning and then extended it to twin support vector machine (TSVM) learning through the RoBoSS-based Twin Support Vector Machine (RoBoTS) framework. The results demonstrated that robust loss design can improve classification performance under noisy, outlier-contaminated, and biomedical data settings.

The Wave loss further strengthened the classification-loss perspective by introducing a smooth and bounded loss with asymmetric penalization. By embedding this loss into SVM and TSVM frameworks, the thesis developed Wave-SVM and Wave-TSVM. These models showed that bounded, differentiable, and asymmetric losses can reduce the adverse effect of

unreliable samples while retaining optimization stability. Together, the RoBoSS and Wave loss developments establish that loss functions are not merely technical choices inside a fixed model; rather, they determine which samples dominate the objective, how errors are penalized, and how stable the resulting optimization problem becomes.

For regression, the thesis introduced the HawkEye loss function. Regression tasks require a different robustness mechanism because the response is continuous and errors may be asymmetric or affected by an insensitive region. HawkEye combines boundedness, smoothness, and an insensitive zone, thereby addressing an important gap between non-smooth robust regression losses and smooth losses that do not explicitly ignore negligible deviations. By integrating HawkEye into least-squares support vector regression (LSSVR), the thesis developed HE-LSSVR and solved it through an inverse-free adaptive moment estimation strategy. The experimental study on synthetic, benchmark, time-series, and brain-age datasets demonstrated the value of robust regression loss design for practical prediction problems.

The thesis then addressed class imbalance and sample uncertainty through the RAPID weighting framework. Unlike a fixed class-level reweighting rule, RAPID combines radius-adaptive proximity, local probability information, and class-imbalance correction. This design recognizes that not all samples from the same class are equally reliable: boundary samples may be informative, central mislabeled samples may be harmful, and distant samples may be outliers. By embedding RAPID into least-squares SVM, the thesis developed RAPID-LSSVM-I and RAPID-LSSVM-II using mean-centered and median-centered designs. The results on benchmark datasets and Alzheimer's disease diagnosis showed that adaptive weighting can improve robustness while retaining the computational efficiency of least-squares formulations.

The thesis also extended robust learning to randomized neural networks. Random vector

functional link (RVFL) networks and broad learning systems (BLSs) are attractive because they avoid expensive backpropagation through hidden layers and often reduce training to output-layer optimization. However, conventional squared-error-based training and direct matrix inversion can be sensitive to outliers, noise, and numerical instability. To address this issue, the thesis developed XG-RVFL and Wave-BLS. These frameworks replace squared-error-dominated objectives with robust bounded losses and employ inversion-free optimization strategies. This shows that randomized neural networks can be made more reliable not only by changing their architecture, but also by redesigning their objective functions and solvers.

Finally, the thesis studied randomness at the representation level through copula-aligned weight initialization (CAWI). Standard randomized neural networks typically sample input-to-hidden weights independently, even when the input features exhibit strong statistical dependence. CAWI develops a dependence-aware initialization mechanism by fitting copulas to the empirical dependence structure of the input data and sampling random weights from the learned dependence model. This preserves the attractive closed-form or backpropagation-free nature of randomized neural networks while making their frozen random projections more data-aware. The empirical results across benchmark and biomedical datasets demonstrated that dependence-aware initialization can improve performance with only marginal computational overhead.

Overall, the thesis contributes a unified perspective: robust and efficient learning can be achieved by aligning the loss function, sample weighting, optimization strategy, and random representation mechanism with the statistical difficulties present in the data. The developed methods preserve the interpretability and efficiency of hyperplane-based models and randomized neural networks, while improving their ability to handle noise, class imbalance, and randomness.

9.2 Major Findings

Several conclusions can be drawn from the investigations conducted in this thesis.

1. **Loss-function design has a direct and measurable effect on robustness.** The RoBoSS, Wave, HawkEye, and XG losses demonstrate that boundedness, smoothness, sparsity, asymmetry, and insensitive-zone behavior can be combined in different ways to control the influence of unreliable samples. A carefully designed loss can reduce the effect of outliers and noisy samples without requiring a complete redesign of the underlying learning paradigm.

2. **Robust losses are transferable across model families.** The thesis demonstrates the developed losses within SVM, TSVM, LSSVR, RVFL, and BLS settings. This indicates that the proposed loss functions are not restricted to the particular paradigms used in this thesis. Their mathematical properties can be exploited directly, or after suitable modification, in other machine learning models where empirical-risk minimization or gradient-based optimization is available.

3. **Adaptive sample weighting is essential for class-imbalanced and uncertain data.** The RAPID framework shows that class imbalance should not be treated only as a global class-frequency issue. A reliable weighting scheme must also consider local reliability, distance from representative class regions, and possible label ambiguity. The dual-center construction further shows that the choice of class representative can influence robustness, particularly when the data distribution is skewed or contaminated by outliers.

4. **Inversion-free optimization improves the scalability and numerical reliability of robust models.** Several proposed formulations replace direct matrix inversion or

quadratic programming with iterative gradient-based optimization. This is important for large-scale, noisy, or ill-conditioned data, where matrix inversion can become computationally expensive or numerically unstable.

5. **Randomized neural networks benefit from both robust objectives and data-aware randomization.** The results of XG-RVFL, Wave-BLS, and CAWI show that randomized neural networks should not be viewed as fixed random-feature machines. Their performance depends strongly on the loss used to train the output layer and on the statistical structure used to generate the hidden representation.

6. **Biomedical applications require robustness at multiple levels.** The biomedical experiments provide application-level predictive evidence rather than uniform clinical validation. Chapter 6 evaluates imbalance-aware diagnosis using both accuracy and F-measure, whereas the biomedical analyses in Chapters 3, 4, 5, and 8 rely mainly on accuracy or point estimates, and some small subgroups lack fold-level uncertainty estimates. The collective finding is therefore that robustness to noise, imbalance, and representation variability can improve predictive behavior under the reported protocols; prospective external validation, class-sensitive measures, and uncertainty intervals are required before clinical conclusions are drawn.

9.3 Future Directions

The findings of this thesis open several directions for future research. These directions arise naturally from the scope of the present work: most hyperplane-based formulations target binary classification, several robust losses include data-dependent shape parameters, the randomized neural network contributions mainly focus on shallow architectures, and CAWI presently evaluates candidate copula families empirically. Addressing these aspects

can further broaden the applicability and practical reliability of the proposed frameworks.

9.3.1 Extending Robust Loss Functions to Other Learning Paradigms

The proposed loss functions are not restricted to the specific models considered in this thesis. RoBoSS, Wave, HawkEye, and XG losses can be used directly or modified for other machine learning frameworks. For example, they can be integrated with kernel machines, neural networks, matrix learning models, ensemble methods, and large-margin metric learning. Their boundedness can help reduce the influence of extreme observations, while their smoothness can support gradient-based optimization. In deep learning, these losses may be particularly useful in settings with label noise, medical-image uncertainty, imbalanced disease cohorts, and weak supervision.

Future work may also study new variants of these losses. For instance, one may design losses that combine boundedness with adaptive asymmetry, class-dependent margins, or sample-dependent saturation levels. Such extensions could allow the loss function to adapt its penalization behavior according to uncertainty estimates, local neighborhood structure, or class imbalance.

9.3.2 Direct Multiclass Extensions of SVM and TSVM Models

The developed SVM and TSVM models are primarily designed for binary classification. A natural future direction is to develop direct multiclass versions of RoBoSS-SVM, RoBoTS, Wave-SVM, Wave-TSVM, and RAPID-LSSVM. Instead of reducing multiclass learning to several binary subproblems, a direct formulation could learn multiple separating functions jointly. Such a framework may better capture inter-class relationships and may avoid inconsistencies that arise in pairwise decomposition methods.

For TSVM-based models, multiclass extension is especially interesting because each

class can be represented by a class-specific proximal hyperplane. Robust bounded losses could be used to control the influence of samples from competing classes, while adaptive weights could correct class imbalance across all classes simultaneously. Developing efficient solvers for such formulations would be an important technical challenge.

9.3.3 Automatic Hyperparameter and Loss-Shape Adaptation

Several proposed frameworks rely on parameters that control loss shape, boundedness, asymmetry, neighborhood size, or weighting flexibility. Instead of selecting these parameters through exhaustive search, future work can develop adaptive strategies that tune them during training. Possible directions include bilevel optimization, Bayesian optimization, gradient-based hyperparameter learning, and validation-free criteria based on local data complexity.

For weighting methods such as RAPID, the flexibility parameter and neighborhood size can be made data-adaptive. For example, the neighborhood size may vary across regions of the feature space according to local density, and the class-imbalance correction may be adjusted using uncertainty-aware estimates rather than global class counts alone. This would make the weighting mechanism more responsive to heterogeneous data distributions.

9.3.4 Robust Deep and Ensemble Randomized Neural Networks

The robust loss-based randomized neural network frameworks developed in this thesis can be extended from shallow RVFL and BLS architectures to deep RVFL and deep ensemble RVFL networks. Deep randomized architectures can construct hierarchical representations by stacking random feature mappings across multiple layers. However, deeper random representations can also amplify instability when the random weights are poorly aligned with the data or when the output objective is sensitive to noisy samples.

Future work may integrate RoBoSS, Wave, XG, or related robust losses into deep RVFL and deep ensemble RVFL frameworks. Such models can combine the representational strength of deeper randomized structures with the robustness of bounded loss functions. In addition, inversion-free optimization can be used to avoid numerical difficulties in layer-wise or ensemble-level training. A further direction is to study how robust losses affect diversity and stability across ensemble members.

9.3.5 Automatic Copula Selection for CAWI

In CAWI, different copula families capture different dependence patterns. Gaussian copulas capture symmetric dependence, t-copulas can represent tail dependence, and Archimedean copulas such as Clayton, Frank, and Gumbel capture different asymmetric dependency structures. Since datasets vary in their dependence behavior, a single copula family may not be universally optimal.

A useful extension is to develop an automatic copula-selection algorithm. Such an algorithm could estimate goodness-of-fit measures, information criteria, tail-dependence statistics, or validation-free dependence scores to identify the most suitable copula for a given dataset. Once the appropriate copula family is selected, CAWI can initialize the randomized neural network using only that family instead of fitting and evaluating all candidates. This would reduce computational overhead and make CAWI more practical for large-scale applications.

An even broader direction is to allow feature groups to use different copula families. Many real-world datasets contain heterogeneous feature blocks, where some groups may exhibit symmetric dependence and others may exhibit tail or asymmetric dependence. Group-wise or vine-copula-based CAWI could capture such complex dependencies more accurately.

9.3.6 Theory for Robust and Randomized Learning

The thesis provides theoretical support for several proposed models through properties such as boundedness, smoothness, classification calibration, influence behavior, and generalization analysis. Future research can deepen this theoretical foundation. For robust loss functions, one can study tighter excess-risk bounds, robustness under label-noise models, and convergence guarantees for non-convex bounded objectives. For randomized neural networks, one can investigate how dependence-aware initialization affects approximation capacity, stability, and generalization.

A particularly important theoretical question is how the interaction between random features and robust losses shapes the learned function. Robust losses reduce the influence of unreliable samples, while random features determine the geometry in which those samples are represented. Understanding this interaction may lead to principled guidelines for choosing randomization schemes, loss parameters, and model width.

9.4 Concluding Perspective

The contributions of this thesis collectively show that robust learning is not obtained from a single design choice. It requires coordinated attention to how errors are penalized, how samples are weighted, how optimization is performed, and how representations are initialized. By developing robust losses, adaptive weighting schemes, inversion-free randomized learning methods, and copula-aligned initialization, this thesis provides a set of tools for building learning systems that are more stable under noise, class imbalance, and randomness.

The broader message is that robustness and efficiency need not be competing objectives. With appropriate mathematical design, it is possible to improve resistance to unreliable

data while retaining simple and computationally efficient learning frameworks. The methods developed here therefore provide a foundation for future research on reliable machine learning models that can operate effectively in uncertain, imbalanced, and data-limited environments.

Bibliography

[1] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, (1999).

[2] T. Hastie, R. Tibshirani, J. Friedman, Overview of supervised learning, in: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, (2009), 9–41.

[3] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, Machine Learning 110 (2021) 457–506.

[4] P. J. Huber, Robust estimation of a location parameter, The Annals of Mathematical Statistics 35 (1964) 73–101.

[5] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Advances in Neural Information Processing Systems 9 (1997) 155–161.

[6] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

[7] I. Steinwart, A. Christmann, Support vector machines, Springer Science & Business Media, (2008).

[8] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (1999) 293–300.

[9] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, (2002).

[10] Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 905–910.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[12] C.-F. Lin, S.-D. Wang, Fuzzy support vector machines, IEEE Transactions on Neural Networks 13 (2002) 464–471.

[13] R. Batuwita, V. Palade, FSVM-CIL: fuzzy support vector machines for class imbalance learning, IEEE Transactions on Fuzzy Systems 18 (2010) 558–571.

[14] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[15] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press Cambridge, (2016).

[16] Y.-H. Pao, G.-H. Park, D. J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (1994) 163–180.

[17] G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[18] C. P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE Transactions on Neural Networks and Learning Systems 29 (2017) 10–24.

[19] C. P. Chen, Z. Liu, Broad learning system: A new learning paradigm and system without going deep, in: 2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE, (2017), 1271–1276.

[20] Q. Shi, R. Katuwal, P. N. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognition 117 (2021) 107978.

[21] A. Sklar, Fonctions de répartition à n dimensions et leurs marges, Publications de l'Institut de Statistique de l'Université de Paris 8 (1959) 229–231.

[22] R. B. Nelsen, An Introduction to Copulas, 2 ed., Springer, New York, (2006).

[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[24] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1940) 86–92.

[25] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.

[26] P. L. Bartlett, M. I. Jordan, J. D. McAuliffe, Convexity, classification, and risk bounds, Journal of the American Statistical Association 101 (2006) 138–156.

[27] X. Huang, L. Shi, J. A. Suykens, Support vector machine classifier with pinball loss, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2013) 984–997.

[28] K.-W. Chang, C.-J. Hsieh, C.-J. Lin, Coordinate descent method for large-scale L2-loss linear support vector machines, Journal of Machine Learning Research 9 (2008) 1369–1398.

[29] M.-Z. Liu, Y.-H. Shao, C.-N. Li, W.-J. Chen, Smooth pinball loss nonparallel support vector machine for robust classification, Applied Soft Computing 98 (2021) 106840.

[30] D. Makmuang, W. Ratiphaphongthon, R. Wangkeeree, Smooth support vector machine with generalized pinball loss for pattern classification, The Journal of Supercomputing 79 (2023) 11684–11706.

[31] Y. Ma, Q. Zhang, D. Li, Y. Tian, LINEX support vector machine for large-scale classification, IEEE Access 7 (2019) 70319–70331.

[32] A. Parsian, S. Kirmani, Estimation under Linex loss function, in: Handbook of Applied Econometrics and Statistical Inference, CRC Press, (2002), 75–98.

[33] J. P. Brooks, Support vector machines with the ramp loss and the hard margin loss, Operations Research 59 (2011) 467–479.

[34] D. Liu, Y. Shi, Y. Tian, Ramp loss nonparallel support vector machine for pattern classification, Knowledge-Based Systems 85 (2015) 224–233.

[35] L. Yang, H. Dong, Support vector machine with truncated pinball loss and its application in pattern recognition, Chemometrics and Intelligent Laboratory Systems 177 (2018) 89–99.

[36] X. Shen, L. Niu, Z. Qi, Y. Tian, Support vector machine classifier with truncated pinball loss, Pattern Recognition 68 (2017) 199–210.

[37] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal Loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, (2017), 2980–2988.

[38] Z. Zhang, M. R. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: Advances in Neural Information Processing Systems, volume 31, Curran Associates, (2018).

[39] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, J. Bailey, Symmetric cross entropy for robust learning with noisy labels, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, (2019), 322–330.

[40] E. Amid, M. K. Warmuth, R. Anil, T. Koren, Robust bi-tempered logistic loss based on bregman divergences, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, (2019).

[41] S. Gonzalez, R. Miikkulainen, Improved training speed, accuracy, and data utilization through loss function optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, (2020), 1–8. doi:10.1109/CEC48606.2020.9185777.

[42] S. Gonzalez, R. Miikkulainen, Optimizing loss functions through multivariate Taylor polynomial parameterization, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, (2021), 305–313. doi:10.1145/3449639.3459277.

[43] L. Tang, Y. Tian, C. Yang, P. M. Pardalos, Ramp-loss nonparallel support vector regression: robust, sparse and scalable approximation, Knowledge-Based Systems 147 (2018) 55–67.

[44] K. Wang, P. Zhong, Robust non-convex least squares loss function for regression with outliers, Knowledge-Based Systems 71 (2014) 290–302.

[45] Y. Ye, et al., Robust support vector regression with generic quadratic nonconvex ε-insensitive loss, Applied Mathematical Modelling 82 (2020) 235–251.

[46] X. Liang, Z. Zhang, Y. Song, L. Jian, Kernel-based online regression with canal loss, European Journal of Operational Research 297 (2022) 268–279.

[47] S. Fu, Y. Tian, L. Tang, Robust regression under the general framework of bounded loss functions, European Journal of Operational Research 310 (2023) 1325–1339.

[48] D. Wen, M. S. Levy, BLINEX: A bounded asymmetric loss function with application to bayesian estimation, Communications in Statistics-Theory and Methods 30 (2001) 147–153.

[49] J. Tang, B. Liu, S. Fu, Y. Tian, G. Kou, Advancing robust regression: Addressing asymmetric noise with the BLINEX loss function, Information Fusion 110 (2024) 102463.

[50] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, W. A. Stahel, Robust Statistics: The Approach Based on Influence Functions, John Wiley & Sons, New York, (1986).

[51] J. T. Barron, A general and adaptive robust loss function, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2019), 4331–4339.

[52] M.-M. Zhou, L. Li, Y.-L. Lu, Fuzzy support vector machine based on density with dual membership, in: 2009 International Conference on Machine Learning and Cybernetics, volume 2, IEEE, (2009), 674–678.

[53] Q. Fan, Z. Wang, D. Li, D. Gao, H. Zha, Entropy-based fuzzy support vector machine for imbalanced datasets, Knowledge-Based Systems 115 (2017) 87–99.

[54] X. Tao, et al., Affinity and class probability-based fuzzy support vector machine for imbalanced data sets, Neural Networks 122 (2020) 289–307.

[55] M. Sajid, A. K. Malik, M. Tanveer, P. N. Suganthan, Neuro-fuzzy random vector functional link neural network for classification and regression problems, IEEE Transactions on Fuzzy Systems 32 (2024) 2738–2749.

[56] M. Sajid, M. Tanveer, P. N. Suganthan, Ensemble deep random vector functional link neural network based on fuzzy inference system, IEEE Transactions on Fuzzy Systems 33 (2025) 479–490.

[57] M. Sajid, A. Quadir, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative, GB-RVFL: Fusion of randomized neural network and granular ball computing, Pattern Recognition 159 (2025) 111142.

[58] H. Joe, Dependence Modeling with Copulas, CRC Press, Boca Raton, (2014).

[59] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Review 60 (2018) 223–311.

[60] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (2011) 2121–2159.

[61] Y. Nesterov, A method for unconstrained convex minimization problem with the rate of convergence o (1/k2), in: Dokl. Akad. Nauk. SSSR, volume 269, (1983), 543.

[62] R. L. Iman, J. M. Davenport, Approximations of the critical region of the fbietkan statistic, Communications in Statistics-Theory and Methods 9 (1980) 571–595.

[63] P. B. Nemenyi, Distribution-free multiple comparisons., Princeton University, (1963).

[64] F. Wilcoxon, Individual comparisons by ranking methods, Springer, (1992).

[65] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[66] H. Wang, Y. Shao, Fast generalized ramp loss support vector machine for pattern classification, Pattern Recognition 146 (2024) 109987.

[67] M. Akhtar, M. Tanveer, M. Arshad, Alzheimer's Disease Neuroimaging Initiative, Advancing supervised learning with the wave loss function: A robust and smooth approach, Pattern Recognition 155 (2024) 110637.

[68] Y. Wu, Y. Liu, Robust truncated hinge loss support vector machines, Journal of the American Statistical Association 102 (2007) 974–983.

[69] X. Zhao, S. Fu, Y. Tian, K. Zhao, Asymmetric and robust loss function driven least squares support vector machine, Knowledge-Based Systems 258 (2022) 109990.

[70] F. Dinuzzo, B. Schölkopf, The representer theorem for Hilbert spaces: a necessary and sufficient condition, Advances in Neural Information Processing Systems 25 (2012) 189–196.

[71] P. L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: Risk bounds and structural results, Journal of Machine Learning Research 3 (2002) 463–482.

[72] Y. Feng, Y. Yang, X. Huang, S. Mehrkanoon, J. A. Suykens, Robust support vector machines for classification with nonconvex and smooth losses, Neural Computation 28 (2016) 1217–1247.

[73] S. Mendelson, A few notes on statistical learning theory, in: Advanced Lectures on Machine Learning: Machine Learning Summer School 2002 Canberra, Australia, February 11–22, 2002 Revised Lectures, Springer, (2003), 1–40.

[74] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, Advances in Neural Information Processing Systems 26 (2013) 315–323.

[75] H. Yuan, T. Ma, Federated accelerated stochastic gradient descent, Advances in Neural Information Processing Systems 33 (2020) 5332–5344.

[76] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Networks 12 (1999) 145–151.

[77] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[78] G. C. Cawley, N. L. C. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, Journal of Machine Learning Research 11 (2010) 2079–2107.

[79] J. Derrac, S. Garcia, L. Sanchez, F. Herrera, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult. Valued Log. Soft Comput 17 (2015) 255–287.

[80] D. Dua, C. Graff, UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences, (2017).

[81] R. G. Andrzejak, et al., Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, Physical Review E 64 (2001) 061907.

[82] M. A. Ganaie, A. Kumari, A. K. Malik, M. Tanveer, EEG signal classification using improved intuitionistic fuzzy twin support vector machines, Neural Computing and Applications 36 (2024) 163–179.

[83] F. A. Spanhol, L. S. Oliveira, C. Petitjean, L. Heutte, A dataset for breast cancer histopathological image classification, IEEE Transactions on Biomedical Engineering 63 (2015) 1455–1462.

[84] C. Gautam, et al., Minimum variance-embedded deep kernel regularized least squares method for one-class classification and its applications to biomedical data, Neural Networks 123 (2020) 191–216.

[85] C. Y. Deng, A generalization of the Sherman–Morrison–Woodbury formula, Applied Mathematics Letters 24 (2011) 1561–1564.

[86] M. A. Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, Expert Systems with Applications 36 (2009) 7535–7543.

[87] M. Mohri, Foundations of machine learning, MIT Press, (2018).

[88] M. Tanveer, A. Sharma, P. N. Suganthan, General twin support vector machine with pinball loss function, Information Sciences 494 (2019) 311–327.

[89] Q. Si, Z. Yang, J. Ye, Symmetric LINEX loss twin support vector machine for robust classification and its fast iterative algorithm, Neural Networks 168 (2023) 143–160.

[90] M. Akhtar, M. Tanveer, M. Arshad, RoBoSS: A robust, bounded, sparse, and smooth loss function for supervised learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 47 (2025) 149–160.

[91] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

[92] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2016), 785–794.

[93] M. Khojaste-Sarakhsi, S. S. Haghighi, S. F. Ghomi, E. Marchiori, Deep learning for Alzheimer's disease diagnosis: A survey, Artificial Intelligence in Medicine 130 (2022) 102332.

[94] S. Srivastava, R. Ahmad, S. K. Khare, Alzheimer's disease and its treatment by different approaches: A review, European Journal of Medicinal Chemistry 216 (2021) 113320.

[95] B. Richhariya, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative, An efficient angle-based universum least squares twin support vector machine for classification, ACM Transactions on Internet Technology (TOIT) 21 (2021) 1–24.

[96] Q. Wang, Y. Ma, K. Zhao, Y. Tian, A comprehensive survey of loss functions in machine learning, Annals of Data Science 9 (2022) 187–212.

[97] M. Tanveer, T. Rajani, R. Rastogi, Y.-H. Shao, M. A. Ganaie, Comprehensive review on twin support vector machines, Annals of Operations Research 339 (2024) 1223–1268.

[98] T. Sypherd, et al., A tunable loss function for robust classification: Calibration,

landscape, and generalization, IEEE Transactions on Information Theory 68 (2022) 6021–6051.

[99] A. Kumari, M. Akhtar, M. Tanveer, M. Arshad, Diagnosis of breast cancer using flexible pinball loss support vector machine, Applied Soft Computing 157 (2024) 111454.

[100] T. Tieleman, G. Hinton, Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning 4 (2012) 26–31.

[101] A. K. Malik, M. A. Ganaie, M. Tanveer, P. N. Suganthan, Alzheimer's disease diagnosis via intuitionistic fuzzy random vector functional link network, IEEE Transactions on Computational Social Systems 11 (2024) 4754–4765.

[102] L. Zhang, P. N. Suganthan, A comprehensive evaluation of random vector functional link networks, Information Sciences 367 (2016) 1094–1105.

[103] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, (2000).

[104] A. Porsteinsson, R. Isaacson, S. Knox, M. Sabbagh, I. Rubino, Diagnosis of early Alzheimer's disease: clinical practice in 2021, The Journal of Prevention of Alzheimer's Disease 8 (2021) 371–386.

[105] S. L. Goldenberg, G. Nir, S. E. Salcudean, A new era: artificial intelligence and machine learning in prostate cancer, Nature Reviews Urology 16 (2019) 391–403.

[106] J. S. Angarita-Zapata, A. D. Masegosa, I. Triguero, A taxonomy of traffic forecasting regression problems from a supervised learning perspective, IEEE Access 7 (2019) 68185–68205.

[107] M. Akhtar, M. Tanveer, M. Arshad, GL-TSVM: A robust and smooth twin support vector machine with guardian loss function, in: International Conference on Pattern Recognition, Springer, (2024), 63–78.

[108] M. Tanveer, et al., Machine learning techniques for the diagnosis of Alzheimer's disease: A review, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16 (2020) 1–35.

[109] B. Richhariya, M. Tanveer, EEG signal classification using universum support vector machine, Expert Systems with Applications 106 (2018) 169–182.

[110] M. Tanveer, et al., Deep learning for brain age estimation: A systematic review, Information Fusion 96 (2023) 130–143.

[111] M. A. Ganaie, M. Tanveer, I. Beheshti, Brain age prediction using improved twin SVR, Neural Computing and Applications 36 (2024) 53–63.

[112] C. C. Onyekwena, et al., Support vector machine regression to predict gas diffusion coefficient of biochar-amended soil, Applied Soft Computing 127 (2022) 109345.

[113] M. Beniwal, A. Singh, N. Kumar, Forecasting long-term stock prices of global indices: A forward-validating genetic algorithm optimization approach for support vector regression, Applied Soft Computing 145 (2023) 110566.

[114] H.-C. Zhang, Q. Wu, F.-Y. Li, Application of online multitask learning based on least squares support vector regression in the financial market, Applied Soft Computing 121 (2022) 108754.

[115] A. Zahariev, P. Angelov, S. Zarkova, Estimation of bank profitability using vector error correction model and support vector regression, Economic Alternatives 28 (2022) 157–170.

[116] D. M. Hawkins, Identification of outliers, volume 11, Springer, (1980).

[117] X. Shen, G. C. Tseng, X. Zhang, W. H. Wong, On ψ-learning, Journal of the American Statistical Association 98 (2003) 724–734.

[118] A. K. Malik, R. Gao, M. A. Ganaie, M. Tanveer, P. N. Suganthan, Random vector functional link network: recent developments, applications, and future directions, Applied Soft Computing 143 (2023) 110377.

[119] L. Zhang, M. Wang, M. Liu, D. Zhang, A survey on deep learning for neuroimaging-based brain disorder analysis, Frontiers in Neuroscience 14 (2020) 779.

[120] A. Benussi, et al., Classification accuracy of blood-based and neurophysiological markers in the differential diagnosis of Alzheimer's disease and frontotemporal lobar degeneration, Alzheimer's Research & Therapy 14 (2022) 155.

[121] V. R. Sampathkumar, Adiag: Graph neural network based diagnosis of Alzheimer's disease, arXiv preprint arXiv:2101.02870 (2021).

[122] H. Sun, A. Wang, W. Wang, C. Liu, An improved deep residual network prediction model for the early diagnosis of Alzheimer's disease, Sensors 21 (2021) 4182.

[123] Alzheimer's Association, 2023 Alzheimer's disease facts and figures, Alzheimer's & Dementia 19 (2023) 1598–1695.

[124] F. Rauf, et al., Denseinceps115: a novel network-level fusion framework for Alzheimer's disease prediction using mri images, Frontiers in Oncology 14 (2024) 1501742.

[125] S. Abadal, et al., Graph neural networks for electroencephalogram analysis: Alzheimer's disease and epilepsy use cases, Neural Networks 181 (2025) 106792.

[126] W. Peng, et al., Fusion of brain imaging genetic data for Alzheimer's disease diagnosis and causal factors identification using multi-stream attention mechanisms and graph convolutional networks, Neural Networks 184 (2025) 107020.

[127] B. Richhariya, M. Tanveer, A. H. Rashid, Alzheimer's Disease Neuroimaging Initiative, Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE), Biomedical Signal Processing and Control 59 (2020) 101903.

[128] K. Shihabudheen, G. N. Pillai, Recent advances in neuro-fuzzy system: A survey, Knowledge-Based Systems 152 (2018) 136–162.

[129] Y. Wang, S. Wang, K. K. Lai, A new fuzzy support vector machine to evaluate credit risk, IEEE Transactions on Fuzzy Systems 13 (2005) 820–831.

[130] E. L. Crow, M. M. Siddiqui, Robust estimation of location, Journal of the American Statistical Association 62 (1967) 353–389.

[131] P. J. Rousseeuw, Tutorial to robust statistics, Journal of Chemometrics 5 (1991) 1–20.

[132] A. Kumari, M. Tanveer, C. T. Lin, Class probability and generalized bell fuzzy twin SVM for imbalanced data, IEEE Transactions on Fuzzy Systems 32 (2024) 3037–3048.

[133] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge University Press, (2004).

[134] K. S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is nearest neighbor meaningful?, in: Proceedings of the 7th International Conference on Database Theory, Springer, (1999), 217–235. doi:10.1007/3-540-49257-7_15.

[135] C. Xu, C.-T. Li, Y. Hu, C. P. Lim, D. Creighton, Deep learning techniques for video instance segmentation: A survey, Pattern Recognition 167 (2025) 111763.

[136] M. Tanveer, et al., Ensemble deep learning for Alzheimer's disease characterization and estimation, Nature Mental Health 2 (2024) 655–667.

[137] M. Tanveer, et al., Fuzzy Deep Learning for the Diagnosis of Alzheimer's Disease: Approaches and Challenges, IEEE Transactions on Fuzzy Systems 32 (2024) 5477–5492.

[138] A. B. Mokal, B. Gupta, A deep learning approach for effective classification of fingerprint patterns and human behavior analysis, Pattern Recognition 163 (2025) 111439.

[139] L. Yi, D. Lv, D. Liu, S. Li, R. Liu, Self-training-transductive-learning broad learning system (STTL-BLS): A model for effective and efficient image classification, Pattern Recognition 156 (2024) 110747.

[140] W. Cao, X. Wang, Z. Ming, J. Gao, A review on neural networks with random weights, Neurocomputing 275 (2018) 278–287.

[141] L. Zhang, P. N. Suganthan, A survey of randomized algorithms for training neural networks, Information Sciences 364 (2016) 146–155.

[142] B. B. Hazarika, D. Gupta, Random vector functional link with ε-insensitive Huber loss function for biomedical data classification, Computer Methods and Programs in Biomedicine 215 (2022) 106622.

[143] M. Akhtar, R. Mishra, M. Tanveer, M. Arshad, Advancing RVFL networks: Robust classification with the HawkEye loss function, in: Neural Information Processing, volume 15288 of *Lecture Notes in Computer Science*, (2025), 226–240.

[144] M. Akhtar, M. Tanveer, M. Arshad, HawkEye: A robust loss function for regression with bounded, smooth, and insensitive zone characteristics, Applied Soft Computing 176 (2025) 113118.

[145] F. R. Hampel, Contributions to the theory of robust estimation, University of California, Berkeley, (1968).

[146] M. J. Black, A. Rangarajan, On the unification of line processes, outlier rejection, and robust statistics with applications in early vision, International Journal of Computer Vision 19 (1996) 57–91.

[147] J. Zhang, H. Yang, Bounded quantile loss for robust support vector machines-based classification and regression, Expert Systems with Applications 242 (2024) 122759.

[148] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, Cambridge University Press, (2014).

[149] M. A. Ganaie, M. Tanveer, P. N. Suganthan, Minimum variance embedded random vector functional link network, in: Neural Information Processing: 27th International Conference, ICONIP 2020, Proceedings, Part V 27, Springer, (2020), 412–419.

[150] M. Sajid, A. Quadir, M. Tanveer, Wave-RVFL: A randomized neural network based on Wave loss function, in: Neural Information Processing, Springer Nature Singapore, Singapore, (2025), 242–257.

[151] S. Feng, C. P. Chen, Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification, IEEE Transactions on cybernetics 50 (2018) 414–424.

[152] M. Sajid, A. K. Malik, M. Tanveer, Intuitionistic fuzzy broad learning system: Enhancing robustness against noise and outliers, IEEE Transactions on Fuzzy Systems 32 (2024) 4460–4469.

[153] W. Deng, et al., Robust dual-model collaborative broad learning system for classification under label noise environments, IEEE Internet of Things Journal 12 (2025) 21055–21067.

[154] A. Voulodimos, et al., Deep learning for computer vision: A brief review, Computational Intelligence and Neuroscience 2018 (2018) 7068349.

[155] D. W. Otter, et al., A survey of the usages of deep learning for natural language processing, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 604–624.

[156] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Transactions on Neural Networks 6 (1995) 1320–1329.

[157] M. Akhtar, et al., Towards robust and inversion-free randomized neural networks: The XG-RVFL framework, Pattern Recognition 172 (2026) 112711.

[158] M. Tanveer, M. A. Ganaie, A. Bhattacharjee, C. T. Lin, Intuitionistic fuzzy weighted least squares twin SVMs, IEEE Transactions on Cybernetics 53 (2023) 4400–4409.