Multi-label Classification using Non-Iterative Learning and Deep Learning based approaches

Ph.D. Thesis

By

Vikas Chauhan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

DECEMBER 2022

Multi-label Classification using Non-Iterative Learning and Deep Learning based approaches

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in partial fulfillment of the requirements for

the award of the degree

of

DOCTOR OF PHILOSOPHY

By

Vikas Chauhan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

DECEMBER 2022

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Multi-label Classification using Non-Iterative Learning and Deep Learning based approaches** in the partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from July 2017 to July 2022 under the supervision of Prof. Aruna Tiwari, Professor, Indian Institute of Technology Indore, Indore, India.

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.

11 July 2022

Signature of the Student with Date

(Vikas Chauhan)

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

Signature of Thesis Supervisor with Date

(Prof. Aruna Tiwari)

Vikas Chauhan has successfully given his Ph.D. Oral Examination held on 2 December 2022

2.12.2022

Signature of Thesis Supervisor with Date

(Prof. Aruna Tiwari)

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. At first, I would like to thank my supervisor **Prof. Aruna Tiwari**, who was a constant source of inspiration during my work. Without her constant guidance and research directions, this research work could not be completed. Her continuous support and encouragement has motivated me to remain streamlined in my research work.

I am thankful to **Prof. Kapil Ahuja** and **Dr. M. Tanveer**, my research progress committee members for taking out some valuable time to evaluate my progress all these years. Their valuable comments and suggestions helped me to improve my work at various stages. I am especially grateful to Dr. Kapil Ahuja, and Dr. M. Tanveer for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives. I am also thankful to **Prof. Anupam Shukla** and **Dr. Pradip Swarnakar** who introduced me in the field of research by his proper guidelines during my masters.

My sincere acknowledgement and respect to **Prof. Suhas Joshi**, Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

I would like to appreciate the fine company of my dearest colleagues and friends especially, Om Prakash Patel, Chandan Gautam, Niranjan Joshi, Sahaj Khandelwal, Akhilesh Mohan Srivastava, Vivek Singh Baghel, Shivvrat Arya and Rituraj. I am also thankful to undergraduate students who have also supported me in my research work. I would like to express my heartfelt respect to my parents for their love, care and support they have provided to me throughout my life.

Finally, I am thankful to all who directly or indirectly contributed, helped and supported me.

Vikas Chauhan

To my family and friends

Abstract

Multi-label classification is a generalization of traditional classification techniques where classes are not mutually independent and multiple classes are assigned to each instance. Multi-label classification is applied in various domains such as image classification, functional annotation of protein sequences, and text categorization. The main objective of this thesis is to propose multi-label classifiers using non-iterative learning and deep learning approaches. We propose non-iterative multi-label classifiers with an adaptive threshold to consider the correlation among labels. The non-iterative approaches provide a fast and accurate solution for multi-label classification. We propose non-iterative randomization-based neural networks for multi-label classification. These multi-label neural networks are named as Multi-label Broad Learning System (ML-BLS), Multi-label Fuzzy Broad Learning System (ML-FBLS), Multi-label Random Vector Functional Link Network (ML-RVFL), and Multi-label Kernelized Random Vector Functional Link Network (ML-KRVFL). The output weights of these neural networks are computed using pseudoinverse. At the output layer, multi-label classification is performed by using an adaptive threshold function. The computation of output weights using pseudoinverse retains the faster computation power of these algorithms compared to iterative learning algorithms. The adaptive threshold function used in the proposed approach can consider the correlation among the output labels and the whole dataset for threshold computation. Five multi-label evaluation metrics, hamming loss, ranking loss, one error, coverage, and average precision, evaluate the proposed multi-label neural networks on 12 benchmark datasets of various domains such as text, image, and genomics. The ML-KRVFL provides the overall best Friedman rankings on five evaluation metrics, followed by ML-RVFL, ML-FBLS, and ML-BLS, respectively. Based on the experimentation results, the proposed ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL perform better than other relevant multi-label approaches in this mentioned order. The non-iterative approaches use the inverse to compute the parameters of algorithms, and it is cumbersome to compute the inverse for large data. Hence, we have proposed deep learning-based multi-label classifiers

for image and sequential data. The deep learning approaches are capable of processing a large amount of raw data such as images and primary protein sequences. In the deep network of GCN, nodes of GCN become similar, and it becomes difficult to differentiate them. This scenario is known as over-smoothing. We propose the Graph Convolution Network based multi-label classifier $MLGCN_{pairnorm}$ for the image domain, which incorporates a normalization-based technique pairnorm to tackle the over-smoothing problem. Further, we have extended this approach for zero-shot learning in the image domain named as $ML-ZSLPGCN$, which is able to classify the images corresponding to unseen labels. The proposed deep learning-based approaches in the image domain are evaluated on MS-COCO, VOC-PASCAL, and NUS-WIDE data based on average precision, recall, and F1 score.

The $MLGCN_{pairnorm}$ and $ML-ZSLPGCN$ are unable to classify the sequential data; hence we have proposed a deep learning method for the sequential domain based on heuristic rules to classify the proteins into their functional classes using a convolution neural network. It is able to handle the data imbalance problem. The protein sequences belong to the functional classes based on the structure of their sequences and the label distribution of this data is imbalanced. The greatest challenge with multi-label classification is to handle the data imbalance, which appears due to variance in frequencies of the labels in the data. This data imbalance is dealt with weight modulation in the loss function to influence the learning process. The annotation task of protein sequences into corresponding functional classes is multi-label in nature. This proposed approach works on the primary structure of protein sequences which considers the relationship between motifs and amino acids. The sequential data in this work is taken from the SwissProt subset of the UniProt database for the experiments. The proposed approach also takes into account the amino acid locations in the protein sequence. The proposed approach considers the affinity information between amino acids and motifs. Along with achieving high performance in the classification of protein sequences, we propose a heuristic approach to improve the precision and recall of the individual functional classes.

In this thesis, all the proposed non-iterative and deep learning multi-label classifi-

cation approaches are applicable to various domains such as image, medical imaging, and bioinformatics.

Keywords: Multi-label classification, Random Vector Functional Link Network, Broad Learning System, Deep Learning, Non-iterative learning, Graph Convolution Network, Over-smoothing, Data imbalance.

List of Publications

A. Published

A1. In Refereed Journals

1. **Vikas Chauhan** and Aruna Tiwari, *Pairnorm based Graphical Convolution Network for zero-shot multi-label classification*, Engineering Applications of Artificial Intelligence, (Elsevier), 114, 2022: 105012. (**IF: 7.802**) **DOI: https://doi.org/10.1016/j.engappai.2022.105012**

2. **Vikas Chauhan** and Aruna Tiwari, *Randomized neural networks for multilabel classification*, Applied Soft Computing, (Elsevier), 115, 2022: 108184. (**IF: 8.263**) **DOI: https://doi.org/10.1016/j.asoc.2021.108184**

3. **Vikas Chauhan**, Aruna Tiwari, Boppudi Venkata, and Vislavath Naik, *Tackling over-smoothing in Multilabel image classification using graphical convolution neural network*, Evolving Systems, (Springer), 2022 (**IF: 2.347) DOI: https://doi.org/10.1007/s12530-022-09463-z**

4. **Vikas Chauhan**, Aruna Tiwari, Niranjan Joshi, and Sahaj Khandelwal, *Multi-label classifier for protein sequence using heuristic-based deep convolution neural network*, Applied Intelligence, (Springer), 52(3), 2021: 2820-2837. (**IF: 5.019**) **DOI: https://doi.org/10.1007/s10489-021-02529-6**

A2. In Refereed Conferences

1. **Vikas Chauhan**, Aruna Tiwari, and Shivvrat Arya, *Multi-Label classifier based on Kernel Random Vector Functional Link Network*, International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom, 2020, pp. 1-7, 2020 **DOI: https://doi.org/10.1109/IJCNN48605.2020.9207436**

2. **Vikas Chauhan** and Aruna Tiwari, *On the Construction of Hierarchical Broad Learning Neural Network: An Alternative Way of Deep Learning*, 2018 IEEE Sym-

posium Series on Computational Intelligence (SSCI), Bangalore, India, 2018, pp. 182-188, 2018 **DOI: https://doi.org/10.1109/SSCI.2018.8628786**

B. Journal publication (other than thesis)

In Refereed Journals

1. **Vikas Chauhan**, Raghvendra Gaur, Aruna Tiwari, and Anupam Shukla, *Real-time BigData and Predictive Analytical Architecture for healthcare application*, Sadhana, Springer, 44(12), 2019: 1-12. (**IF: 1.214**) **DOI: https://doi.org/10.1007/s12046-019-1220-z**

Contents

Abstract i

List of Publications v

List of Figures xi

List of Tables xiii

List of Abbreviations and Acronyms xvii

1 Introduction 1

1.1 Background 3

1.2 Motivation 5

1.3 Objectives 7

1.4 Thesis Contributions 7

1.5 Organization of The Thesis 11

2 Literature Survey and Research Methodology 15

2.1 Multi-Label Classification 15

2.2 Traditional Approaches for Multi-Label Classification 17

2.3 Non-iterative Learning for Multi-Label Classification 19

2.3.1 Pseudoinverse and Ridge Regression 21

2.3.2 The Kernel Trick 22

2.4 Deep Learning approaches for Multi-Label Classification 23

2.4.1 Graph Convolution Network (GCN) 26

2.4.2 Over-smoothing problem in GCN 28

2.5 Zero-Shot Learning (ZSL) in Multi-Label Classification 29
2.6 Multi-Label Classification for Sequential Data 31
2.6.1 Related Work for Multi-Label Classification of Protein Sequences 32
2.6.2 Primary Protein Sequence Structural Representation 34
2.7 Datasets for Multi-Label Classification 36
2.7.1 Multi-Label Datasets for Non-Iterative Algorithms 36
2.7.2 Multi-Label Datasets for Deep Learning Algorithms 38
2.8 Performance Evaluation Metrics 40
2.8.1 Evaluation Metrics for Non-Iterative Multi-Label Algorithms . . 41
2.8.2 Evaluation Metrics For Deep Learning based Multi-Label Algorithms 42
2.9 Statistical Tests 44
2.9.1 Wilicoxon Signed Ranksum Test 44
2.9.2 Ablation Study 44

3 Broad Learning based Multi-Label Classification 45

3.1 Introduction 46
3.2 Multi-Label Classifier based on Broad Learning System: ML-BLS . . . 47
3.3 Multi-Label Classifier based on Fuzzy Broad Learning System: ML-FBLS 49
3.4 Adaptative Thredhold for Multi-Label Classification 53
3.5 Experimentation and Results 55
3.5.1 Experimental Setup 57
3.5.2 Results and Performance Evaluation 59
3.6 Statistical Analysis 64
3.7 Analysis of Computational Complexity 65
3.8 Summary 66

4 Random Vector Functional Link Neural Network based Multi-Label Classification 67

4.1 Introduction 68

4.2 Multi-Label Classifier based on Random Vector Functional Link Neural Network: ML-RVFL . . . 68
4.3 Multi-Label Kernelized Random Vector Functional Link Network: ML-KRVFL . . . 71
4.4 Experimentation and Results . . . 73
4.4.1 Experimental Setup . . . 74
4.4.2 Results and Performance Evaluation . . . 75
4.5 Analysis of Computational Complexity . . . 85
4.6 Summary . . . 85

5 Graph Convolution Network based Multi-Label Classification 87
5.1 Introduction . . . 88
5.2 Multi-Label Classifier based on GCN to Tackle Over-smoothing Problem: $\boldsymbol{MLGCN_{pairnorm}}$ . . . 91
5.3 Experiments and Results . . . 94
5.3.1 Results on MS-COCO dataset . . . 95
5.3.2 Results on VOC 2007 dataset . . . 96
5.3.3 Ablation Studies . . . 97
5.4 Summary . . . 99

6 Zero-Shot Learning for Multi-Label Classification using Graph Convolution Network 101
6.1 Introduction . . . 102
6.2 Preliminaries . . . 103
6.2.1 Multi-Label Zero-Shot Learning . . . 103
6.2.2 Attention Regional Embedding . . . 104
6.3 Proposed ZSL Multi-Label Classifier using GCN: $\boldsymbol{ML-ZSLPGCN}$ 107
6.3.1 Image representation module . . . 107
6.3.2 Label-aware module . . . 108
6.3.3 Multi-Label Graph Convolution Network module . . . 109

6.4 Experiments and Results . . . 113
6.4.1 Ablation Study . . . 116
6.5 Summary . . . 117

7 Heuristic based Deep Learning Multi-Label Classifier for Sequential Protein Data 119

7.1 Introduction . . . 119
7.2 UniProt Data Representation and its Filtering . . . 121
7.2.1 Filtering of UniProt data . . . 122
7.2.2 Encoding of protein sequences . . . 123
7.3 Heuristic based Deep Learning approach for Multi-Label Protein Sequences . . . 123
7.4 Experimentation and Results . . . 127
7.4.1 Experimental Results and Discussion . . . 127
7.4.2 Computational Complexity . . . 139
7.5 Summary . . . 140

8 Conclusions and Future Work 141

8.1 Summary of Research Achievements . . . 142
8.2 Future Research Directions . . . 144

Bibliography 146

List of Figures

1.1 Illustration of Multi-label Classification 2
1.2 Categorization of multi-label algorithms 4
1.3 Flow diagram of thesis work 13

2.1 Random Vector Functional Link Neural Network 20
2.2 Visualization of data in lower and higher dimensions 23
2.3 Label representation in GCN in the forward pass 25
2.4 correlation between labels 26
2.5 Structure of Alanine and Glycine amino acid 35
2.6 Multi-label dataset categorization 36
2.7 Sample images of MS-COCO dataset 38
2.8 Sample images of NUS-WIDE dataset 39
2.9 Sample images from VOC 2007 dataset 40

3.1 Broad Learning System 47
3.2 Multi-Label Broad Learning System Neural Network 48
3.3 Multi-Label Fuzzy broad learning system 51

4.1 Multi-label Random Vector Functional Link Neural Network 69

5.1 Example of label dependencies in multi-label classification 90
5.2 Flow diagram of pair norm 92
5.3 Proposed classifier with pair norm 93
5.4 Results provided by the $MLGCN_{pairnorm}$ approach for all classes of VOC 2007 dataset 97

5.5 Results provided by the $MLGCN_{pairnorm}$ approach for top 3 classes of VOC 2007 dataset 98

6.1 Attention Regional Embedding 105
6.2 Complete $ML-ZSLPGCN$ approach using pairnorm 107
6.3 Flow diagram of pairnorm 110
6.4 F1 score performance vs. label correlation threshold τ 116
6.5 F1 score vs number of epochs 116

7.1 Embedding of ACDAD partial proteing sequence with size $W \times L$ . . . 122
7.2 Heuristic based system architecture 124
7.3 Label Frequency Distribution 134
7.4 Precision and Recall vs Label Frequency 134
7.5 Plot of Recall for individual Labels for Models - 10,11 and 12. 136
7.6 Plot of Precision for individual Labels for Models - 10,11 and 13. . . . 136
7.7 Plot of Recall for individual Labels for Models - 11,13 and 14. 137
7.8 Plot of Precision for individual Labels for Models - 11,13 and 15. 137
7.9 Plot of Precision for individual Labels for Models - 11,14 and 16. 138

List of Tables

2.1 Multi-label classification definition example 16
2.2 Binary Relevance for multi-label classification 16
2.3 Classifier chain for multi-label classification 17
2.4 Label Powerset approach for multi-label classification 17
2.5 Essential Amino Acids 34
2.6 Multi-label data-set description 37
2.7 Raw data acquired from UniProt 39
2.8 Multi-label data-set description 43

3.1 Optimal parameter settings for multi-label classifiers 58
3.2 Comparison results performed on corel5k dataset 59
3.3 Comparison results performed on emotions dataset 60
3.4 Comparison results performed on enron dataset 60
3.5 Comparison results performed on medical dataset 60
3.6 Comparison results performed on scene dataset 61
3.7 Comparison results performed on yeast dataset 61
3.8 Comparison results performed on bibtex dataset 61
3.9 Comparison results performed on rcv1v2s1 dataset 62
3.10 Comparison results performed on rcv1v2s2 dataset 62
3.11 Comparison results performed on rcv1v2s3 dataset 62
3.12 Comparison results performed on rcv1v2s4 dataset 63
3.13 Comparison results performed on rcv1v2s5 dataset 63
3.14 Average of performance measures on multi-label datasets 63

3.15 Statistics of p-values obtained from Wilcoxon's signed rank test for hamming loss . . . 64
3.16 Statistics of p-values obtained from Wilcoxon's signed rank test for ranking loss . . . 64
3.17 Statistics of p-values obtained from Wilcoxon's signed rank test for one error . . . 64
3.18 Statistics of p-values obtained from Wilcoxon's signed rank test for coverage . . . 65
3.19 Statistics of p-values obtained from Wilcoxon's signed rank test for average precision . . . 65

4.1 Optimal parameter settings for multi-label classifiers . . . 74
4.2 Comparison results performed on corel5k dataset . . . 75
4.3 Comparison results performed on emotions dataset . . . 76
4.4 Comparison results performed on enron dataset . . . 76
4.5 Comparison results performed on medical dataset . . . 77
4.6 Comparison results performed on scene dataset . . . 77
4.7 Comparison results performed on yeast dataset . . . 78
4.8 Comparison results performed on bibtex dataset . . . 78
4.9 Comparison results performed on rcv1v2s1 dataset . . . 79
4.10 Comparison results performed on rcv1v2s2 dataset . . . 79
4.11 Comparison results performed on rcv1v2s3 dataset . . . 80
4.12 Comparison results performed on rcv1v2s4 dataset . . . 80
4.13 Comparison results performed on rcv1v2s5 dataset . . . 80
4.14 Average of performance measures on multi-label datasets . . . 81
4.15 Average Friedman rank for multi-label classification algorithms . . . 81
4.16 Statistics of p-values obtained from Wilcoxon's signed rank test for hamming loss . . . 83
4.17 Statistics of p-values obtained from Wilcoxon's signed rank test for ranking loss . . . 83

4.18 Statistics of p-values obtained from Wilcoxon's signed rank test for one error . . . 84
4.19 Statistics of p-values obtained from Wilcoxon's signed rank test for coverage . . . 84
4.20 Statistics of p-values obtained from Wilcoxon's signed rank test for average precision . . . 84
4.21 Running time in seconds for training procedure . . . 84

5.1 Parameter values of the classifier . . . 94
5.2 The comparison results for all labels on MS-COCO dataset . . . 96
5.3 The comparison results for top 3 labels on MS-COCO dataset . . . 97
5.4 Comparison of mAP with state-of-the-art approaches on VOC 2007 dataset . . . 98
5.5 The effect of over-smoothing on MS-COCO dataset . . . 99
5.6 The effect of over-smoothing on VOC 2007 datset . . . 99

6.1 Dataset description . . . 113
6.2 Parameters used in the proposed approach $ML-ZSLPGCN$ . . . 114
6.3 Comparison results on MS-COCO dataset . . . 114
6.4 Comparison results for NUS81 dataset . . . 115
6.5 Ablation study for $ML-ZSLPGCN$ . . . 117

7.1 Network architecture and parameters for model 0 . . . 128
7.2 Network architecture and parameters for model 1 . . . 128
7.3 Network architecture and parameters for model 2 . . . 129
7.4 Network architecture and parameters for model 3 . . . 129
7.5 Network architecture and parameters for model 4 . . . 130
7.6 Network architecture and parameters for model 5 . . . 130
7.7 Network architecture and parameters for model 6 . . . 131
7.8 Network architecture and parameters for model 7 . . . 131
7.9 Network architecture and parameters for model 8 . . . 132

7.10 Network architecture and parameters for model 9 132
7.11 Network architecture and parameters for model 10 133
7.12 Summary of the results 133
7.13 Results provided by Model 12 135
7.14 Results provided by Model 13 135
7.15 Results provided by Model 14 135
7.16 Results provided by Model 15 135
7.17 Results provided by Model 16 135
7.18 Summary of the results using heuristic approach 138
7.19 Parameter setting for the comparative approaches 139
7.20 comparison of prediction performance across various models 139

List of Abbreviations and Acronyms

RVFL Random Vector Functional Link Network

BLS Broad Learning System

GCN Graph Convolution Network

ELM Extreme Learning Machine

FBLS Fuzzy Broad Learning System

ZSL Zero Shot Learning

HBLS Hierarchical Broad Learning System

KELM Kernel Extreme Learning Machine

KRR Kernel Ridge Regression

SVM Suport Vector Machine

BP-MLL Backpropagation for Multi-Label Learning

$\boldsymbol{ML - GCN_{pairnorm}}$ Multi-Label Graphical convolutional Network usin pairnorm

$\boldsymbol{ML - ZSLPGCN}$ Pairnorm based Multi-Label Zero-Shot Learning using Graphical convolutional Network

ARE Attention Regional Embedding

LSTM Long Short Term Memory

biLSTM bidirectional Long Short Term Memory

GRU Gated Recurrent Unit

SVD Singular Value Decomposition

PIR Protein Identification Resource

PPI Protein Protein Interaction

1D-CNN 1-Dimensional Convolution Neural Network

AUC Area Under Curve

SA Subset Accuracy

CNN Convolution Neural Network

Chapter 1

Introduction

In traditional supervised learning, each real-world object is denoted by a single instance, and it is associated with a single label[1, 2]. One fundamental assumption in traditional supervised learning is that one input instance belongs to only one label. Traditional supervised learning is successful in many tasks, but in the real world, there are many supervised tasks that do not fit in the assumption of traditional supervised learning as real-world objects are associated with multiple labels simultaneously [3, 2]. For example, a newspaper article can be associated with the labels of history and crime at the same time, or a single image can contain the label of the sky, birds, and trees [3]. In the music domain, a piece of music may belong to mozart, austria, and classical simultaneously. To consider the association of multiple labels with one instance, one direct solution is to assign a set of proper labels to the input instance. Following this consideration, the paradigm of multi-label classification emerges [4].

Multi-label classifier classifies an input instance that is associated with more than one output label simultaneously. A multi-label classifier finds a set of appropriate labels for an input data instance. Multi-label classification is considered as a generalization of multi-class or binary classification [2]. In multi-label classification, the number of associated labels is not known prior, so the label space grows exponentially. For example, a label space with 20 labels grows to more than one million(i.e., 2^{20}). To tackle this large label space, it is essential to consider the correlation among labels [4, 1].

Figure 1.1: Illustration of Multi-label Classification

The earliar research of multi-label classification was mainly focused on text categorization [5, 6, 7] but in the present time it has been widely applied to the various problem of bioinformatics [3, 8, 9], automatic annotation of multimedia content [10, 11, 12], rule mining [13, 14], web mining [15, 16], and tag recommandation [17, 18].An example of multi-label classification is illustrated in Figure 1.1 as it contains more than one label, such as window, sky, tree, and grass. Due to the non-prior information about associated labels and the association of multiple labels with a single instance, multi-label classification is considered harder than traditional multi-class classification. The following examples provide a simple understanding of multi-label classification in various domains.

(i) **Image recognition:** There are multiple objects available in a single image. The task to identify the multiple objects in an image is known as multi-label classification in image recognition[19, 20].

(ii) **Music Classification:** Music has various types of categories based on moods and interests, such as sad, happy, romantic, and patriotic. A single piece of music can belong to multiple classes, so this classification fits in the multi-label

classification [21].

(iii) **Text categorization:** In the case of a newspaper article, a single text can fit into the economics and politics topics simultaneously. Multi-label classification in the text-domain is considered to find the simultaneous topics for the given text [5, 6].

(iv) **Functional Genomics Categorization:** Genome sequences are created by proteins, and these proteins are created by various amino acids. Alphabetic characters denote the essential amino acids for better representation. The properties of the genome are characterized by these amino acids, and each amino acid is responsible for various characteristics of the genome, such as color, age, and strength. So genome classification is considered as multi-label classification [22, 23].

1.1 Background

As discussed an instance in the multi-label classification is associated with more than one class/label [24, 2]. Due to this, multi-label classification is considered a generalization of multi-class classification. In literature, a backpropagation based multi-label classifier (BP-MLL) [3] is considered the first neural network based classifier. This is an iterative classifier, and it changes the loss function of the traditional back-propagation operation. Rank-SVM is another multi-label classifier that is iterative in nature and it is based on the Support Vector Machine(SVM) [8]. These approaches have a larger training procedure due to their iterative nature. Due to a large number of labels, it becomes difficult to converge these algorithms [24]. The training procedure of iterative algorithms is reduced by developing the non-iterative algorithms for multi-class and multi-label classification.

The multi-label classification algorithms are categorized into two categories [2]. The first category is known as Problem transformation, and the second is known as Algorithm adaptation. This categorization of multi-label classification is demonstrated

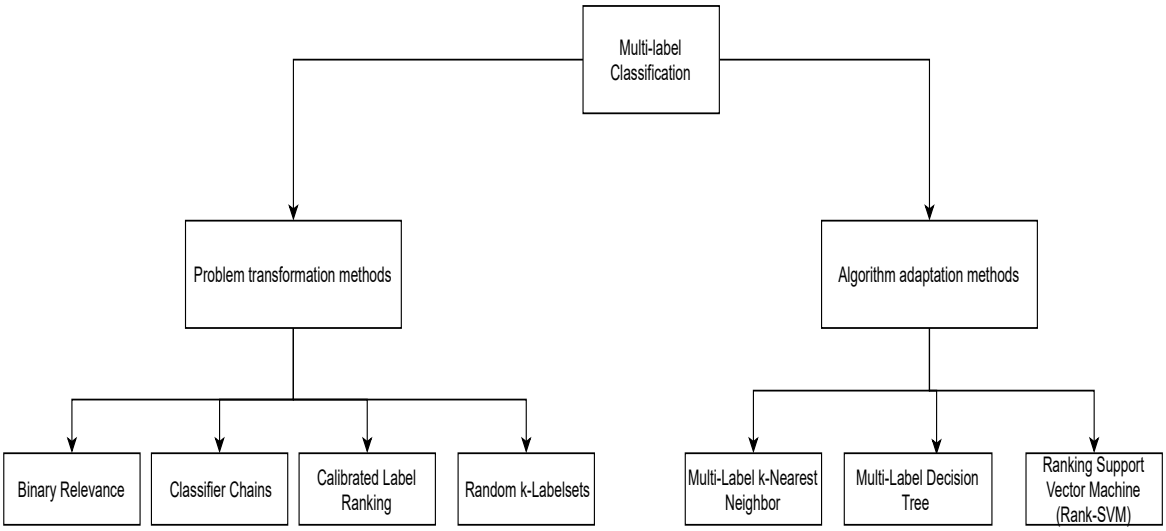


Figure 1.2: Categorization of multi-label algorithms

in Figure 1.2. The categories of problem transformation and algorithm adaptation are defined as follows:

(i) **Problem transformation:** This category of algorithms tackles multi-label learning problems by transforming them into other well-established learning scenarios. The key philosophy of problem transformation methods is to fit data to the algorithm.

(ii) **Algorithm adaptation:** This category of algorithms tackle multi-label learning problems by adapting popular learning techniques to deal with multi-label data directly. The key philosophy of algorithm adaptation methods is to fit the algorithm to data.

The problem transformation methods are complex to converge and algorithm adaptation approaches make changes in the traditional algorithms to adapt these algorithms for multi-label classification. In recent times the multi-label adaptation of backpropagation algorithm, support vector machine, and k nearest neighbor are adapted for multi-label as BP-MLL [3], RankSVM [8], and Multi-label k nearest neighbor (ML-KNN) [25] respectively. We have proposed multi-label classifiers based on non-iterative and deep learning approaches, which come under the algorithm adaptation category. The non-iterative algorithms focus on fast and accurate solutions to classification prob-

lems. The existing non-iterative algorithms such as Extreme Learning Machine(ELM) [26], Random Vector Functional Link Network(RVFL) [27, 28], Broad Learning System (BLS) [29] and Hierarchical Broad Learning System(HBLS) [30] are able to provide a fast and optimal solution for the traditional classification problems. However, the scope of these approaches is limited to binary classification and multi-class classification. Hence we propose the multi-label adaptation of traditional non-iterative approaches RVFL [27] and BLS [29]. The non-iterative classifiers use the matrix inverse to compute the parameters of the algorithms. The computation of parameters using the inverse becomes cumbersome for large data. The non-iterative algorithms take the data in the form of preprocessed features and these approaches are unable to process a large amount of raw data. To overcome these limitations, we have also proposed multi-label classifiers using deep learning. The deep learning approaches are capable of working with a large amount of raw data; for example, the Imagenet database is trained on the size of approximately 150 gigabytes of images. In the deep learning based approaches, we present the multi-label classifiers for the image domain and sequential domain [31, 32]. The motivation behind this thesis work is described in the next section.

1.2 Motivation

As discussed earlier, research in recent times has shown that iterative learning based approaches such as BP-MLL[3] and RankSVM [8] have performed very well in multi-label classification. However, iterative approaches are computationally expensive; hence there is a need to develop non-iterative approaches for multi-label classification [33]. In recent years, few researchers have developed the non-iterative learning model for multi-label classification such as ML-ELM [33] and ML-KELM [24]. In these approaches, there is scope for improvement in performance on the basis of various evaluation measures such as hamming loss, ranking loss, and one error. It is difficult for a single algorithm to perform optimum on multiple evaluation measures[3]. The RVFL and BLS are the non-iterative Single Layer Feedforward Neural networks

(SLFN), which use the random initialization of weight and bias of the network and compute the output layer weights using the pseudoinverse. The RVFL and BLS are non-iterative algorithms and proposed for traditional multi-class classification. We have proposed the multi-label adaptation of RVFL and BLS approaches. Further, both iterative and non-iterative approaches for multi-label classification need preprocessed features, and these algorithms are not able to perform well enough for a large number of labels[34]. In the case of a large number of labels, at the output layer of iterative and non-iterative classifiers, the difference among labels becomes insignificant. The non-iterative algorithms use matrix operation pseudoinverse or ridge regression to compute the parameters, so these approaches become difficult to use for a large amount of data. To overcome these issues, we have proposed deep learning based multi-label classifiers that are able to perform multi-label classification on a large amount of raw data, which is available in huge amounts. In the case of a large no. of labels, the difference among labels becomes obscured and capturing the dependency between labels also becomes difficult for deep learning approaches.

To resolve the problem of dealing with a large number of labels, Graph Convolution Network (GCN) has been used to represent the labels as a feature vector [35]. These feature vectors contain the semantic properties of labels and are known as semantic embeddings or label embeddings. A large number of labels can be represented as nodes in GCN; hence GCN is able to provide scalability to the multi-label classifier in terms of the number of labels. The nodes in the GCN become similar after the convolution operation if more than two hidden layers are present in GCN. It means GCN is not able to differentiate the labels in case of more than two hidden layers [36, 37]. This scenario is known as the over-smoothing problem in GCN. We have incorporated a normalization scheme pairnorm to tackle the over-smoothing problem in GCN for multi-label classification [38] in image domain. Further, we have extended this GCN based multi-label classifier to consider zero-shot learning where instances of some labels are not available during training. To classify the sequential data, deep learning approaches have been explored in multi-label classification. We have enhanced the performance of the deep learning based classifier [39] using a heuristic approach

for sequential data consisting of the primary protein sequences. Overall this thesis addresses various issues of existing non-iterative and deep learning based algorithms for multi-label classification.

1.3 Objectives

In this thesis, we aim to achieve the following objectives:

(i) To develop non-iterative multi-label classifiers using a broad learning system.

(ii) To develop a method based on a random vector functional link network, which can perform multi-label classification with better performance based on evaluation metrics such as coverage and average precision.

(iii) To develop the deep learning based multi-label classifier which can classify a large amount of raw image data with a large number of associated labels.

(iv) To enable the zero-shot learning for a large amount of raw image data where instances of some labels are not present during training for classification.

(v) To provide a multi-label classifier for the sequential data which is suitable for the classification of bioinformatics related multi-label classification problems.

1.4 Thesis Contributions

The notable contribution of the research work in multi-label classification is to propose the algorithms which are capable of considering the label dependencies. The other contributions consider reducing the training time and improving the evaluation metrics. A brief overview of our research contributions is provided below, which are further detailed in the form of the chapters.

Contribution I:

The state-of-the-art multi-label algorithms heavily depend on the datasets and domain of the data. Existing iterative multi-label classifiers take a considerable huge

time for convergence during training. In order to overcome this problem, in this thesis, we explore the non-iterative multi-label classifiers, which are fast and accurate for multi-label classification. We have proposed randomized non-iterative Multi-label Broad Learning System (ML-BLS) and Multi-label Fuzzy Broad Learning System (ML-FBLS), which are based on broad learning system [29]. ML-FBLS is the neuro-fuzzy architecture in which the mapped features of a broad learning system are replaced by fuzzy subsystems. In these approaches at the output layer, multi-label classification is performed by using an adaptive threshold function. The computation of output weights using pseudoinverse retains the faster computation power of these algorithms compared to iterative learning algorithms. The adaptive threshold function used in the proposed approach can consider the correlation among the output labels and the whole dataset for threshold computation. Five multi-label evaluation metrics evaluate the proposed multi-label neural networks on 12 benchmark datasets of various domains such as text, image, and genomics.

Contribution II:

The broad learning system based multi-label classifiers have scope to improve the coverage and average precision performance measures. We propose another non-iterative approach Random Vector Functional Link (RVFL) network based simple and effective multi-label classifiers by adding an adaptive threshold at the output layer. The estimation of neurons in hidden layers is needed in RVFL. The kernelized based approaches can be used with RVFL based approach for the multi-label classification, which resolves the estimation of neurons in the hidden layer in the RVFL. We propose two approaches; the first is based on an adaptation of RVFL for multi-label classification ML-RVFL, and another is its kernelized version for multi-label classification ML-KRVFL, which are the most efficient non-iterative classifiers proposed in this thesis. In the comparative analysis of the proposed four non-iterative multi-label classifiers, the ML-KRVFL provides the overall best Friedman rankings on five evaluation metrics, followed by ML-RVFL, ML-FBLS, and ML-BLS, respectively. Based on the experimentation results, the proposed ML-KRVFL, ML-RVFL, ML-FBLS, and ML-BLS perform better than other relevant multi-label approaches.

Contribution III:

The performance of the multi-label classifiers and training procedure depends on the number of labels available in the dataset. In the case of a large number of labels, it becomes cumbersome to provide the solution of multi-label classification for non-iterative approaches. The non-iterative approaches depend on inverse computation, and for large-size datasets, it becomes difficult to compute the inverse of the data matrix. The importance of the graph convolution network in multi-label classification has grown in recent years due to its label embedding representation capabilities. The graph convolution network is able to capture the label dependencies using the correlation between labels. However, the graph convolution network suffers from an over-smoothing problem when the layers are increased in the network. Over-smoothing makes the nodes indistinguishable in the deep graph convolution network. We have proposed a normalization technique to tackle the over-smoothing problem in the graph convolution network for multi-label classification. The proposed approach is an efficient multi-label object classifier based on a graph convolution neural network that tackles the over-smoothing problem. The proposed approach normalizes the output of the graph such that the total pairwise squared distance between nodes remains the same after performing the convolution operation. The proposed approach outperforms the existing state-of-the-art approaches based on the results obtained from the experiments performed on MS-COCO and VOC2007 datasets. The experimentation results show that pairnorm mitigates the effect of over-smoothing in the case of using a deep graph convolution network.

Contribution IV:

In the multi-label classifier based on GCN, there is a possibility that for some labels, instances are not available during training. This case is known as Zero-shot learning (ZSL) [40]. Zero-shot learning transfers the knowledge from the seen labels available during training to the unseen labels [40]. To consider the ZSL for multi-label classification, we have included an attention mechanism and label-aware module in the $MLGCN_{pairnorm}$. The proposed approach $ML-ZSLPGCN$ uses the label features obtained from the images during training for seen labels and semantic embedding for

the unseen labels. The $ML-ZSLPGCN$ first creates the features corresponding to the images, and the label-aware module creates the feature vector of the labels corresponding to the seen labels using the attention region embedding. A graph convolution network takes the feature vector of seen labels during training and semantic word embedding for the unseen labels as input and learns the classifier. The proposed approach uses a pairnorm-based normalization scheme to tackle the over-smoothing problem in the graph convolution network. The experimental results on the NUSWIDE and MS-COCO datasets show that the proposed approach provides significant performance in terms of precision, recall, and F1 score in comparison to state-of-the-art approaches.

Contribution V:

The sequential data contains the sequence of characters, and deep learning approaches need to be developed for the classification of sequential data. In this thesis, we develop a heuristic based deep learning classifier which is able to optimize the performance of the sequential data. The heuristic approach is inspired by harmonic heuristics [41], which is easy to implement with deep learning techniques. The sequential data used in the thesis is in the form of amino acid sequences of the UniProt dataset. Along with achieving high performance in the classification of protein sequences, we propose a heuristic approach to improve the precision and recall of the individual functional classes. The proposed heuristic approach improves the performance and handles the data imbalance problem. The heuristic approach is inspired by harmonic heuristics [41], which is easy to implement with deep learning techniques. The proposed approach is compared with other competitive approaches, and our approach provides better performance metrics in terms of precision, recall, AUC, and subset accuracy. The greatest challenge with multi-label classification is to handle the data imbalance, which appears due to variance in frequencies of the labels in the data. This data imbalance is dealt with weight modulation in the loss function to influence the learning process.

1.5 Organization of The Thesis

This thesis is organized into eight chapters. A summary of each chapter is provided below:

Chapter 1 (Introduction)

This chapter introduces multi-label classification and discusses its application in various domains. Further in this chapter, we have described the background study of multi-label classification, motivation of the thesis, and contribution of this thesis.

Chapter 2 (Literature Survey and Research Methodology)

This chapter describes the detailed literature survey of multi-label classification, non-iterative learning, and deep learning. It also provides a summary of various iterative, non-iterative, and deep learning approaches related to multi-label classification. This chapter also presents the details of performance measures, datasets, and statistical tests used for experimentation throughout the thesis.

Chapter 3 (Broad Learning based Multi-Label Classification)

Broad learning based multi-label classifiers are presented in this chapter. Fuzzy broad learning based multi-label classifier is also discussed later in this chapter, which enhances the performance of the Broad Learning based multi-label classifier. These classifiers are faster than iterative approaches due to their non-iterative nature.

Chapter 4 (Random Vector Functional Link Neural Network based Multi-Label Classification)

This chapter provides the details of the Random Vector Functional Link Neural Network based multi-label classifier. We have also proposed the kernelized version of the multi-label Random Vector Functional Link Neural Network. These classifiers are non-iterative in nature and enhance the performance of broad learning based classifiers with simple architecture.

Chapter 5 (Graph Convolution Network based Multi-Label Classification)

The approaches mentioned in chapters 3 and 4 have various limitations, such as these approaches are non-iterative in nature, so inverse computation of matrix is needed to compute the parameters of the network. For the large datasets and fea-

ture spaces, it is difficult to compute the inverse for parameters computation. The non-iterative approaches take the input data as feature vectors, and these approaches have limited scope for the large raw data samples. The Deep Learning based approaches are useful to process large size raw data. In this chapter, GCN based deep learning approach is used for multi-label classification in the image domain, which is able to classify the raw images. This chapter also proposes the pairnorm technique to tackle the over-smoothing problem in GCN for multi-label classification.

Chapter 6 (Zero-Shot Learning for Multi-Label Classification using Graph Convolution Network)

This chapter describes the zero-shot learning for multi-label classification using GCN. The GCN based approach mentioned in chapter 5 has the limitation that it could not classify the labels whose instances are not available during training. Zero-shot learning makes it possible, so an extension of the chapter 5 for zero-shot learning is described in this chapter.

Chapter 7 (Heuristic based Deep Learning Multi-Label Classifier for Sequential Protein Data)

The non-iterative approaches mentioned in chapters 3 and 4 have limitations that they are unable to classify the large-scale raw data, and the deep learning based multi-label classification approaches mentioned in chapters 5 and 6 have limitations that can not be applied to the sequential raw data such as protein sequences. The properties of protein sequences are multi-label in nature because one sequence of the protein is responsible for multiple functionalities of protein. This is also known as functional annotation of protein sequences [42, 43]. In this chapter, we describe one deep learning based heuristic approach related to the bioinformatics domain, which annotates the protein sequences.

Chapter 8 (Conclusions and Future Work)

This chapter describes the conclusion and contribution of the approaches presented in this thesis. The possible future directions are also discussed in this chapter. The complete flow of work is illustrated in Figure 1.3 on the next page.

Figure 1.3: Flow diagram of thesis work

Chapter 2

Literature Survey and Research Methodology

This chapter provides the literature review of multi-label classification and identifies the research gap in the literature. In this chapter, we divide the literature review into nine sections. The mathematical definition of multi-label classification is discussed in section 2.1. Section 2.2 discusses the literature of traditional multi-label algorithms followed by the discussion of non-iterative algorithms in section 2.3. Section 2.4 provides the details of the literature review related to multi-label classification in the image domain using deep learning algorithms. Section 2.5 provides the details of the literature review related to zero-shot learning in multi-label classification in the image domain. In section 2.6, the literature review of sequential data for multi-label classification is discussed, followed by the discussion of multi-label datasets and evaluation metrics used for multi-label classification in section 2.7 and 2.8 respectively. The details of the statistical tests used in this thesis are discussed in section 2.9.

2.1 Multi-Label Classification

As discussed earlier, multi-label classification is generalization of traditional multi-class classification and more than one labels are associated with an input instance. To define multi-label classification, let the domain of the input instance be denoted by $\mathcal{X} \in R^{N \times M}$ and the output label space is represented by $\mathcal{Y} = 1, 2, ..., Q$. Here Q is the

Table 2.1: Multi-label classification definition example

$\mathcal{X}$	**label** y_1	**label** y_2	**label** y_3	**label** y_4
x_1	1	1	1	0
x_2	1	0	0	1
x_3	1	0	1	0
x_4	0	1	1	0

Table 2.2: Binary Relevance for multi-label classification

$\mathcal{X}$	**label** y_1
x_1	1
x_2	1
x_3	1
x_4	0

$\mathcal{X}$	**label** y_2
x_1	1
x_2	0
x_3	0
x_4	1

$\mathcal{X}$	**label** y_3
x_1	1
x_2	0
x_3	1
x_4	1

$\mathcal{X}$	**label** y_1
x_1	0
x_2	1
x_3	0
x_4	0

total number of classes or labels in the label set. In the given training samples denoted by $T = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ which consist the pair of instances $x_i \in \mathcal{X}$ and its corresponding label $y_i \subseteq \mathcal{Y}$, multi-label classifier provides some optimized evaluation metric based on the output $h : \mathcal{X} \to 2^{\mathcal{Y}}$. Multi-label classifier provides the outputs as a function which provides real values denoted by $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. These real values can be easily transformed to the ranking function $rank_f(.,.)$. This ranking function is further used to separate the set of associated labels from the non-associated labels related to a particular instance. The association of output labels using a multi-label classifier can also be computed by a threshold function $t(.)$. In literature, this threshold function is usually a fixed value or constant function [2, 24]. So, in conclusion, the multi-label classifier provides more than one corresponding output label for an unseen instance. The illustrative example of multi-label classification is shown in Table 2.1, where four instances $\{x_1, x_2, x_3, x_4\} \in \mathcal{X}$ have the corresponding labels entires as 1 in $\{y_1, y_2, y_3, y_4\} \in \mathcal{Y}$ labels. These notations are used throughout the thesis to denote the instances and labels. After the discussion of basic concepts of multi-label classification, we have discussed the traditional multi-label classification approaches in the next section.

Table 2.3: Classifier chain for multi-label classification

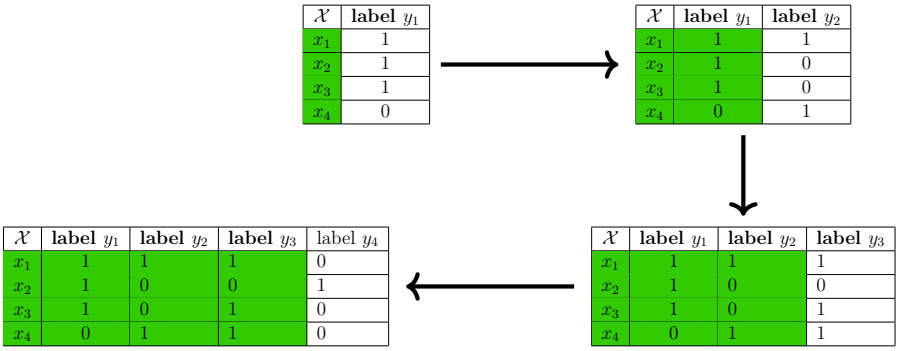


Table 2.4: Label Powerset approach for multi-label classification

$\mathcal{X}$	**Label set**
x_1	1
x_2	2
x_3	3
x_4	4

2.2 Traditional Approaches for Multi-Label Classification

As discussed in chapter 1, problem transformation and algorithm adaptation are two categories of multi-label classification. Binary relevance [44], classifier chain[45], and label powerset [46, 47] are examples of some algorithms that come under the problem transformation category. BP-MLL[3], Rank-SVM[8], and ML-KNN [25]are the example of some algorithms that come under the algorithm adaptation category. In the binary relevance algorithm, independent multi-label classifiers are trained for each label. The results of these independent classifiers are combined after the training procedure in the binary relevance algorithm. Because the binary relevance considers the independence among labels, the correlation among labels is missing in this algorithm.

The binary relevance is shown in Table 2.2 where the corresponding labels of each instance are independent. The classifier chain is another multi-label algorithm that passes the label information between the classifiers using chains. These chains are created by passing the previous classifier as an input to the next classifier. The

performance of the classifier chain is dependent on the order of classifiers that are provided as chains to another classifier during training. In Table 2.3 the classifier chain algorithm is shown where the previous output is provided as an input to find the next label of a particular instance. The inputs are shown in green color, and chains in the classifier chain incorporate label correlation. In the label power set, the sets of associated labels for each input instance are created to denote the associated label set as a single label. Then the multi-label problem is solved as a multi-class problem using the label power set approach. Table 2.4 represents the label powerset which represents the label set as a single label. The label sets shown in Table 2.1 are shown as a particular label in Table 2.4. The construction of the label set is a time-consuming process, and it increases the complexity of the label power set algorithm.

In recent times [3, 24, 48], the main focus is carried on the algorithm adaptation based algorithms in which the multi-class algorithms are being adapted for multi-label classification. In the algorithm adaptation approaches, Boostexter [6] is a famous approach to solve multi-label learning, which assigns and maintains a set of weights over training instances and their corresponding output labels [6]. A Bayesian approach uses a mixture of probabilistic models to generate each document. This mixture of weight and distribution of words in each mixture component is learned using the Expectation-Maximization(EM) algorithm [49]. BP-MLL[3], Rank-SVM [8], and ML-KNN [25] are neural network, SVM, and k-nearest neighbour based multi-label algorithms respectively which considers the label correlation among labels. The main limitation of these algorithms is the long training procedure that exists in both problem transformation and algorithm adaptation based algorithms. The problem of a longer training procedure is resolved by using non-iterative algorithms, which are introduced in the next section.

2.3 Non-iterative Learning for Multi-Label Classification

Non-iterative approaches are useful to reduce the training time of the algorithms with the optimum solution. For the non-iterative algorithms, the concepts of flat networks [50] using multiple mixed activations were proposed in 1989 by Pao et al. These flat networks use the non-iterative procedure to train the network parameters for regression or classification tasks. The classification abilities of flat networks have been the main research interest in recent times, and these are being investigated in the present [51]. An efficient and fast non-iterative neural network to compute the learnable parameters of the neural network using pseudoinverse is known as Random Vector Functional Link Network (RVFL). Sunganthan et al. [28] performed various experiments to confirm the importance and good performance of non-iterative algorithms. A deep version of RVFL, based on stacked autoencoders, introduces the denoising criterion and recovers clean inputs from their corrupted version [52]. A sparse pre-trained version of RVFL known as SP-RVFL has proposed to use a sparse autoencoder with a regularizer using the L_1 norm for unsupervised learning. The SP-RVFL handles the issue with random parameters using the valuable information provided with the input data [53]. The nonlinear expansion of the input vector into a set of orthogonal functions has its advantages over the performance of the other classifiers based on neural networks. The Orthogonal Polynomial Expanded Random Vector Functional Link Neural Network (OPE-RVFLNN) utilizes advantages from the expansion of the input vector and random determination of the input weights [54]. Extreme Learning Machine(ELM) [26] is similar to RVFL[27] and the main difference between ELM and RVFL is that ELM does not have direct links between inputs and output layer and bias is also not present in ELM. As shown in Figure 2.1, the RVFL takes the input samples in the form of the matrix X and feed to the input layer $H1$. The enhancement nodes are created with the randomly initialized weight W_h and $H1$. The output layer is denoted using Y, and in the RVFL, the output weights W are computed using the pseudoinverse. The direct connections denote the original features of input data, and

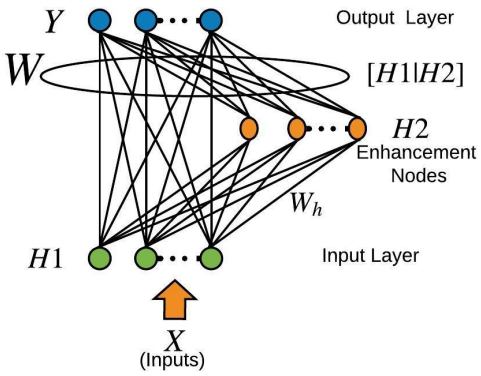


Figure 2.1: Random Vector Functional Link Neural Network

these direct links play an important role in the performance of the neural network and improve the performance of the classifier [55, 56]. The performance of randomization-based approaches has been improved by applying different combinations of RVFL networks in recent times [52]. The features are reused in the form of direct connections in RVFL based structures. Few non-iterative approaches are explored in the multi-label classification, and a lot of scopes are there to explore these approaches in the multi-label classification. The adaptive version of ELM for multi-label classification is known as ML-ELM [57], and it is able to classify streamed multi-label data. Another multi-label classifier ML-KELM based on the kernelized ELM (KELM), was proposed to provide stability to the ELM-based multi-label approaches[24]. In literature the KELM is identical to the Kernel Ridge Regression(KRR) [58]. In this thesis, we have mentioned the multi-label adaptation of KELM as ML-KRR/ML-KELM.

BP-MLL, ML-ELM, and ML-KRR/ML-KELM are neural network based approaches to solve multi-label learning problems. Broad Learning System(BLS) is another neural network that is based on the RVFL [29]. The neural network based classifiers work in a manner that the internal information related to execution is hidden in it. After the training, neural networks work like a black box that is unable to explain

how the data is approximated. Fuzzy-rules-based approaches are used to solve this interpretability-related issue, which imposes the rules for the execution. Interpretable rules can be imposed on algorithms such as fuzzy rules for better understanding[59]. The parameters of fuzzy rules are learned using iterative procedures. The neuro-fuzzy hybrid systems are capable of adapting the fuzzy rules for the training data and finding suitable fuzzy rules without human intervention [60]. The neuro-fuzzy models are universal approximators of functions, and this is one advantage of a hybrid model based on neuro-fuzzy models [61]. Because of these advantages, neuro-fuzzy models have been used in diverse fields of research such as classification [61, 62, 63], regression [64, 65] and time series prediction [66, 67]. RVFL has been used to solve fuzzy nonlinear regression (FNR) problems using both inputs and outputs as trapezoidal fuzzy numbers [68]. The fuzzy broad learning system (FBLS) is a neuro-fuzzy model for multi-class classification [69]. This comprises the advantages of both the neural network and fuzzy system. Both BLS and FBLS are non-iterative in nature, and both are able to solve traditional multi-class problems. The non-iterative algorithms are faster due to the use of pseudoinverse; hence the output layer weights are computed using the pseudoinverse in the BLS. Thus, the discussion of pseudoinverse is given in the next section.

2.3.1 Pseudoinverse and Ridge Regression

Pseudoinverse is considered a generalized inverse, and it can be considered a suitable approach to computing the output layer weights of neural networks. Orthogonalization method, orthogonal projection method, and Singular Value Decomposition(SVD) are some approaches that are used to compute the pseudoinverse [70, 71]. The output weights in the neural network are aimed to reach minimum error, but these weights may not be reached to this due to the ill condition problems. Let H denotes the concatenation of input layer and enhancement node layer which is denoted by mathematical expression

$$H = [H1|H2] \tag{2.1}$$

where H1 and H2 are input layer and enhancement node layer as shown in Figure 2.1. Following mathematical expression denotes the alternative to solve the pseudoinverse

$$\arg\min_{\mathbf{W}} : \|\boldsymbol{HW} - \boldsymbol{Y}\|_v^{\sigma_1} + \lambda\|\boldsymbol{W}\|_u^{\sigma_2} \tag{2.2}$$

where u, v, $\sigma_1 > 0$ and $\sigma_2 > 0$ denote the norm regularization. Above optimization problem can be converted to a convex problem by taking $u = v = \sigma_1 = \sigma_2 = 2$. This solution is an approximation of Moore–Penrose generalized inverse. λ denotes the constraints on W and the value $\lambda = 0$ denotes the solution of original pseudoinverse.

$$\boldsymbol{W} = (\boldsymbol{H^T H} + \lambda \boldsymbol{I})^{-1} \boldsymbol{H}^T \boldsymbol{Y} \tag{2.3}$$

The above formulation is used to compute the output layer weights in the neural network. In neural networks, the number of neurons in the hidden layer is one of the hyperparameters. The kernelized neural network helps to avoid the estimation of hidden layer neurons. Hence in this thesis, we have used the kernel function with RVFL for multi-label classification. The kernel trick is explained in the next section.

2.3.2 The Kernel Trick

Kernel tricks are the core of kernel learning-based methods. It is simply based on the inner product of samples into some new feature space $\boldsymbol{\phi(x)}$. It is defined as follows [72]:

Definition: We say that $\boldsymbol{k(x, y)}$ is a kernel function if and only if there is a feature map such that for all $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$\boldsymbol{k(x, y) = \phi(x).\phi(y)}$$

Here, a kernel function generates a matrix, which is called a kernel or gram matrix. The kernel or gram matrix is a matrix of similarities of pairs of samples. This matrix needs to be symmetric and positive semi-definite. Any function can be treated as a kernel function if it satisfies Mercer condition [72]. Mercer Theorem can be defined as follows:

Mercer Theorem: A symmetric function $\boldsymbol{k(x, y)}$ can be expressed as an inner product $\boldsymbol{k(x, y) = \phi(x).\phi(y)}$ for some $\boldsymbol{\Phi}$ if and only if $\boldsymbol{k(x, y)}$ is positive semidefinite. The Figure 2.2 provides the visualization of data using kernel function in lower

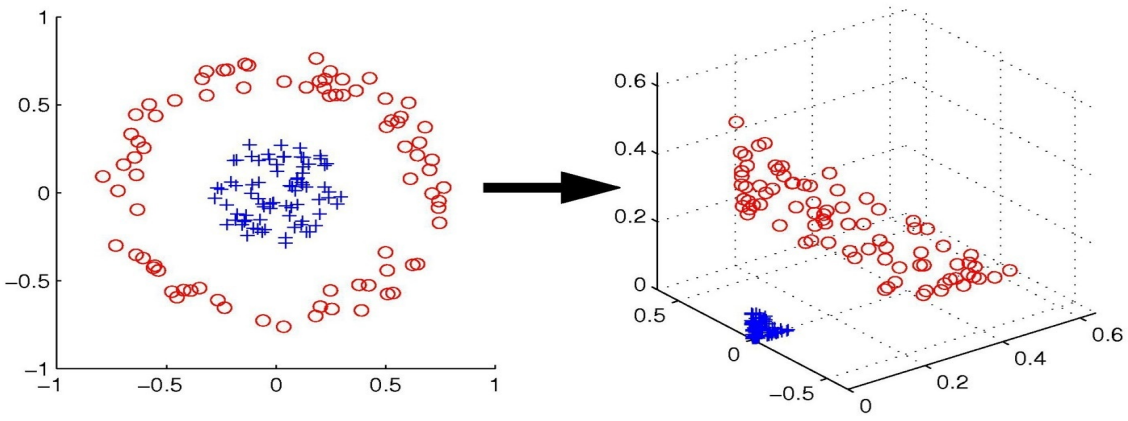


Figure 2.2: Visualization of data in lower and higher dimensions

and higher dimensions. It is not always possible to separate the two classes from each other in the lower dimensional space using a hyperplane. Hence, data is projected into the higher dimensional space so that a hyperplane can easily separate them, and this projection can not be orthogonal.

The non-iterative algorithms use the preprocessed features of the datasets as input. These approaches are unable to process a large amount of raw data because the matrix inverse is used to compute the parameters of the algorithms. The deep learning approaches are able to process a large amount of data and extract the features; hence the discussion of deep learning algorithms for multi-label classification is mentioned in the next section.

2.4 Deep Learning approaches for Multi-Label Classification

The non-iterative approaches use the closed-form solutions such as pseudoinverse and ridge-regression for the computation of the parameters. The pseudoinverse uses

the inverse of a matrix in the computation of matrices, so non-iterative solutions are difficult to use for large-size multi-label datasets of various domains such as image and text sequence. In the image domain, deep learning is being used for the classification of raw image data. The performance of single-label classification in the image domain has been greatly improved using the deep convolution neural networks (CNNs) [31, 32]. Further multi-label classification for the image domain is useful in the field of human attributes, medical diagnosis, and retail checkout recognition[73, 74, 75]. In recent times, rapid progress has been achieved in the development of large image datasets. These datasets are manually labeled, such as PASCAL VOC [76] and MSCOCO [77]. These datasets are available to explore the research using a deep convolution network [78]. For the computer vision-based tasks, a deep-learning based convolution neural network (CNN) can be trained with the softmax function at the output layer[79]. The softmax function at output layer based approaches also considers the multi-label classification as a different single-label classification problem, which uses the ranking loss or cross-entropy loss for the classification [34]. All of these approaches are limited in their scalability as the number of classes or labels grows continuously. In the case of multi-label classification, as the number of classes increases, the distinction between classes becomes unclear. These models are unable to scale their ability for the growing number of labels or classes. The labels are dependent on each other in various manners, such as correlation, prior knowledge related to neighbor nodes, and geometric information of graphs. Among these dependencies, correlation is the key dependency for multi-label classification. The approaches discussed above fail to consider the correlations among labels.

The Recurrent Neural Network (RNN) and probabilistic graph model (PGM) based approaches are proposed to consider the label correlations [80, 81]. PGM approach again suffers from the scalability issue because of the high complexity. RNN predicts the labels in a sequential manner, which is based on some predefined order. Attention mechanism based models implicitly consider the correlations between the regions of an image [82]. The regions of images behave like local correlation, and global correlation requires knowledge beyond a single image. A graph convolution neural network (GCN)

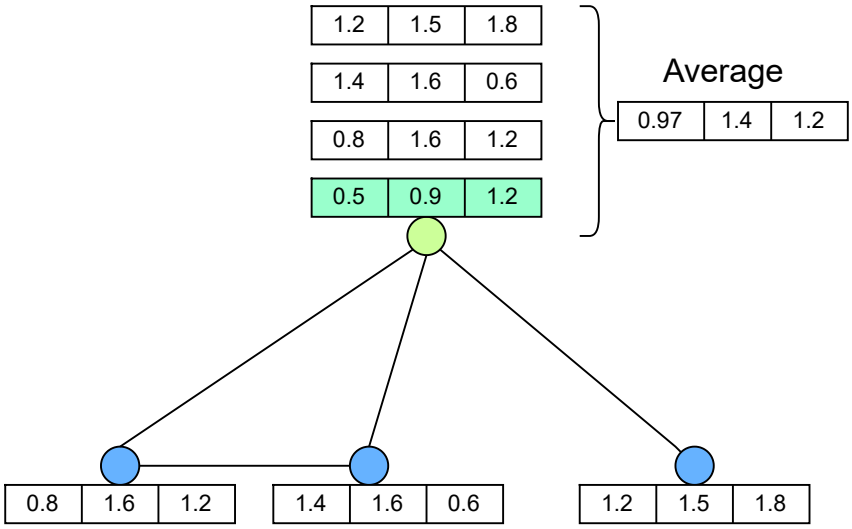


Figure 2.3: Label representation in GCN in the forward pass

based classifier is proposed to resolve the scalability and correlation issues [83, 84]. In the GCN, the nodes represent the feature vectors as embeddings, and edges represent the connection among the nodes. In general, this connection is represented by an adjacency matrix. The insertion of the adjacency matrix in the forward pass equation enables the model to learn the feature representations based on node connectivity. In multi-label classification, these connections are represented by a correlation matrix in place of an adjacency matrix. The resulting GCN can be seen as the first-order approximation of Spectral Graph Convolution in the form of a message-passing network where the information is propagated along the neighboring nodes within the graph. In this case, GCN nodes consider the neighbors and take the value of neighbors and the average embeddings. Figure 2.3 shows that the values of the embedding become similar if the deep GCN is used and the over-smoothing problem occurs due to similar embedding.

Contrary to GCN, in traditional algorithms, the classes or labels are represented by one-hot encoding vectors. But in GCN, the classes or labels are represented as feature vectors. The classes are represented as the feature vectors and these feature vectors contain the characteristics of classes. These feature vectors are created manually or by using the pre-trained vectors. Pretrained vectors are available for label representation, such as Glove [85], FastText [86], GoogleNews[87]. These feature vectors are also

known as label embedding. Using the benefits of label embedding, the nodes can be represented as the nodes of graphs. Each label has its own embedding as a feature vector. For example, as shown in Figure 2.3, nodes represent the feature vectors of labels, and each node is denoted by the average of features of its neighbors. This representation of nodes is used to train the Graph Convolution Network. The details of the Graph Convolution Network are presented in the next section.

2.4.1 Graph Convolution Network (GCN)

Graph Convolution Network is used in semi-supervised learning, which uses the propagation of information from one node to another in a connected graph[35]. In multi-label classification, the labels can be represented as nodes of a graph. These nodes represent the feature vector of individual labels, which is usually the semantic embedding of a particular label. To incorporate zero-shot learning, we use semantic embedding for unseen labels. For the seen labels during training, we use the feature vectors generated by the label-aware module. These semantic vectors and feature vectors are provided as input to the GCN. The GCN is beneficial because the param-

Figure 2.4: correlation between labels

eters of embedding to the generated classifier are shared among labels. This provides advantages for those labels which have weak semantics or have fewer instances. The correlation dependency can also be easily captured by the graph structure by creating the dependency matrix. The graph convolution neural network works on the information flow among nodes. These nodes are correlated as per the properties of the multi-label classification. It is important to consider the correlation among labels effectively.

We use a graph-based structure to capture the correlation dependency among the labels. In the graph, each label is denoted as a node and represents the word embeddings. We propose a GCN-based model which directly maps these embeddings to the image features for the classification, and these mapping parameters are shared across all the classes. The main idea of GCN is to provide feature description A^l and correlation matrix S as an input to the function $f(.,.)$ and learn it for the graph G. The correlation matrix S is used to represent the correlation among nodes. In the GCN, usually, this correlation matrix is predefined based on the adjacency properties of nodes in a graph. The nodes in the GCN graphs represent the labels for multi-label classification. So as per the need for multi-label classification, the correlation matrix S is derived based on the co-occurrence of labels in the dataset. The matrix S represents the semantic embedding among the labels. This matrix is named a co-occurrence matrix because it represents the correlation relationship between labels. If the labels i and j cooccurs together then the probability $P(j|i)$ is not necessarily equals to the probability $P(i|j)$. This can be understand from the Figure 2.4 that $P(cricketer|stumps) = .4$ and $P(stumps|cricketer) = .8$ are different. These correlation behaviors are captured by the matrix S. Let label i occurs n^i times in training set and co-occurrence of two labels i and j is represented by n^{ij}. Using this information, the correlation among label i and j can be denoted by $\frac{n^{ij}}{n^i}$. This computation provides the following matrix, which contains the conditional probabilities

$$\mathbf{P}_i = \frac{n^{ij}}{n^i}, \tag{2.4}$$

The cooccurance measure may be very rare if two labels are not correlated. The cooccurance of labels may be different between test set and training set. So using the value $\frac{n^{ij}}{n^i}$ as cooccurance becomes difficult to generalize. To alleviate these problems, the binary correlation matrix S can be defined as follows

$$\mathbf{S}_{ij} = \begin{cases} 0, \text{ if } \frac{n^{ij}}{n^i} < \tau \\ 1, \text{ if } \frac{n^{ij}}{n^i} \geq \tau \end{cases} \tag{2.5}$$

where S_{ij} represents the binary representation of the correlation and τ denotes the threshold to filter the noisy edges. Every layer in GCN can be written by a non-linear function as

$$\mathbf{A}^{l+1} = f(\mathbf{A}^{l},\ \mathbf{S}). \tag{2.6}$$

The function $f(.,.)$ after applying the convolution operation is denoted by

$$\mathbf{A}^{l+1} = h(\hat{\mathbf{S}}\mathbf{A}^{l}\mathbf{W}^{l}) \tag{2.7}$$

where W^l denotes the learnable transformation matrix, $\hat{S}$ is the normalized version of matrix S, and $h(.)$ represents the nonlinear operation. When the number of layers is increased in the GCN, then the nodes are unable to discriminate among themselves and represent the same values. This scenario is known as an over-smoothing problem. The detail of the over-smoothing problem is described in the next section.

2.4.2 Over-smoothing problem in GCN

The graph convolution neural network based classifiers suffer from the over-smoothing problem [83, 84]. The GCN suffers from the over-smoothing problem when deep structures are used in GCN, as the over-smoothing problem occurs in GCN when the number of layers is increased to create the deep GCN structure. The features of nodes become indistinguishable due to this problem. In other words, for the visual data, over-smoothing refers that the features of labels become similar after applying the convolution operation in the graph. The convolution operators are analogous to laplacian smoothing. When the convolution operation is applied many times to the functions, then the functions converge to similar values. The same scenario happens when the layers are increased in the GCN then the features of nodes converge to similar values due to the convolution operations. This behavior is known as over-smoothing. Over-smoothing makes it difficult to make the GCN deep and decreases the performance of the deep classifier if it is not tackled in the initial phase of network design. In this thesis, we propose a normalization scheme in a multi-label classifier that tack-

les the over-smoothing and reduces the effect of over-smoothing in deep GCN. There is a possibility that instances of some labels are not available during the training of the algorithms. In this case, the transfer of knowledge from the labels corresponding to the available instance is required to the labels corresponding to the unavailable instances. In the case of multi-label classification, we have used zero-shot learning for this knowledge transfer. The details of Zero-shot Learning in multi-label classification are given in the next section.

2.5 Zero-Shot Learning (ZSL) in Multi-Label Classification

For the image domain, deep architectures based on Convolution Neural Network (CNN) have been proposed to assign multiple labels for an image [88, 76, 89, 90]. These approaches are based on the assumption that training data is completely labeled, and during the testing time, all the labels are already present training time as well. In this case, the scope of the transfer of knowledge from seen labels is limited to unseen labels [91]. In real-time scenarios, there is a possibility that instances of some labels are not available or instances are available only for testing or after the training completion. The labels corresponding to these scenarios are known as unseen labels. For these cases, transfer learning is introduced, which transfers the knowledge learned from massive trained data to other data to learn the new classes [92]. Zero-Shot Learning(ZSL) is one learning approach that is used to transfer knowledge from seen to unseen data[92].

ZSL is able to predict those labels whose associated instances are not available during training [92, 91]. For the semantic information, the labels should be in the form of feature vectors that contain the properties of labels. Using the semantic information of labels, the transfer of learning is possible from seen to unseen labels. Using these properties, the instances corresponding to the unseen labels can get the knowledge from the seen labels [92, 93]. The main assumption of ZSL is that labels

should be a numeric vector based on a huge text corpus or a high-dimensional binary vector based on a manually defined object ontology. In this case, each class or label should be embedded in a semantic space[94, 95]. The relationship between seen and unseen labels is established using this semantic label space. Multi-class classification has been focused on using ZSL, and the scope in multi-label classification has been limited[40].

The multi-label classification is explored in various domains with the consideration of label correlation [2, 96]. The scope of ZSL is studied and investigated in multi-label classification [92]. Multi-label classifiers classify the data using missing labels and noisy labels with the help of Sylvester Equation (SMSE)[97]. Some other approaches use the leveraged information from unlabeled and labeled data for multi-label classification [98, 99]. All of these approaches do not consider the emerging new labels and assume that the labels of test data are also present in training data [100, 92].

In the image and vision-related works, the ZSL also has taken attention [101, 102, 103, 104, 105]. Multi-label zero-shot learning assigns more than one label to an input instance where some classes/labels are not available during training. Co-Occurrence Statistics for Zero-Shot Classification (COSTA) uses the co-occurrence statistics and estimates the unseen labels using a weighted combination of seen classes [106]. Transductive multi-label zero-shot learning uses all possible combinations of labels and treats it as ZSL problem solution[107].

ZSL uses the information from the attributes of labels to recognize the new classes or labels. Pairwise rank loss is used to project the visual features on a common embedding space with the help of a bi-linear function [101]. In this way, most of the ZSL is used as a classifier with the visual projection on label space. Some other ZSL-based classifiers use the optimization of compatibility between different perspectives, such as embedding space and a regularization term [108, 103]. All these ZSL-based approaches are explored in multi-class classification and assign only a single label to the input data instance. Zero-shot learning for Multi-label predictors (ZS-MLP) extends the multi-class classifier by considering each label as an independent label. In this case, the complexity of the classifier increases [107]. Due to not considering the

dependency of labels, the performance of ZS-MLP becomes poor with respect to each label set of data.

In the image domain, due to the various combinations of label spaces and mapping between image features and label space, ZSL multi-label spaces have two major issues [106, 109]. The first issue is the limitation of capturing the local features in the image, and the second is ignoring the global label dependencies.

In order to resolve these issues, Ou et al. have proposed the Graph Convolution Network (GCN) based ZSL multi-label classifier in which the labels are represented as the nodes of the graph. This approach contains two modules [40]. The first one is the image module which captures the features of images, and the second is the label GCN which considers the semantic representation of label spaces. The semantic representation of labels is obtained by the pre-trained vectors such as Glove[85] and GoogleNews [87]. GCN is useful to tackle the above issues because global label dependencies are handled using the node representation of the label in a graph. Local features are handled in the image domain using the attention module, and this is combined with GCN for ZSL in multi-label classification.

2.6 Multi-Label Classification for Sequential Data

As discussed in chapter 1, multi-label classification is applied in various domains such as genomics, image, and text. The genomics domain contains the sequential data of raw protein sequences. The sequential data contains the sequence of characters that describes the particular characteristics of the data. Natural language processing, bioinformatics, and genomics use sequential data for particular tasks in respective domains. The non-iterative approaches and deep learning based approaches discussed in section 2.3, 2.4 respectively are unable to process a large amount of raw sequential data. Functional classification using structural information of protein sequences is an important research problem [22, 23]. The problem of functional annotation of protein sequences refers to classifying the protein sequence into various functional categories based on the available structural information. The structural organization

of proteins is done based on the four different levels: primary, secondary, tertiary, and quaternary structure. The sequence of amino acids in the polypeptide chain is referred to as the primary structure of protein [110]. The secondary structure represents the highly regular local sub-structures on the polypeptide backbone chain. There are two main types of secondary substructure classification categories. The first category is known as the α-helix, and the second category is known as the β-a strand or β-sheets. Tertiary structure refers to the three-dimensional geometry of the folded substructures. Quaternary structure is the three-dimensional structure consisting of the aggregation of two or more individual polypeptide chains (subunits) that operate as a single functional unit. Hence, the complexity of the protein structure increases from the primary structure to the quaternary structure. The primary structure level is considered as a basic structure, and the quaternary is the most complex structure. The annotation of proteins with their amino acid sequence as the input and the functional classes as outputs is a multi-label classification problem [43]. Here, the protein can be classified into multiple functional classes at once. It means more than one functional class can be associated or assigned to the protein sequence. A protein sequence can be involved in multiple functionalities; hence it can be classified into multiple classes at once [111]. The multi-label classification of protein sequences and their structural representation is presented in the next subsections.

2.6.1 Related Work for Multi-Label Classification of Protein Sequences

The problem of functional annotation of proteins elevates with a deficiency in the amount of structural information available about the protein [22, 23]. It is possible to find specific distinguishing characteristics of the protein that dictate its main functionalities using the knowledge of protein structures having tertiary and quaternary properties. Whereas only the knowledge of the primary structure of the proteins makes it more challenging to make a correct functional annotation to the macro-molecules. It is similar to predicting the annotation by knowing only the high dimensional rep-

resentation or length of amino acid sequences.

Sequence alignment-based similarity search is one method to annotate protein sequence, which searches the similarity between protein structure and its properly chosen annotated database [43]. Many algorithms, such as the exact Smith-Waterman algorithm[112], BLAST[113], or hidden Markov-model based search [114], can be used for sequence alignment. Once the most similar sequence to the input sequence is found in the database, its functional annotation is assigned to the input sequence. One of the significant problems with this approach is that protein sequences have varying relevance in similarity based on the extent of how conservative their sub-sequences are. Likewise, the 3-dimensional protein structure is more rationed during the evolution when contrasted with the primary structure. As a result, two sequences could have the same three-dimensional geometry and the same functionalities but have different primary structures. Due to this reason, this approach loses the momentum for classification[115]. This drawback makes the need for more sophisticated methods to perform classification than the conventional sequence alignment search.

The primary structure of protein sequence is denoted by the sequence of letters, and each of these letters represents twenty essential amino acids. Hence the protein sequence is considered equivalent to the document where amino acids are comparable to the word in a sentence. Hence the protein sequences can be considered as sequential data where machine learning, natural language processing (NLP), and deep learning algorithms are useful to learn the hidden patterns of these sequences [116, 117, 118, 119]. A very fast-growing field of research in protein classification is the use of Artificial Neural Networks (ANN). The ANNs are frequently used in the image, sound processing, and classification problems [120, 121, 122]. Specifying an appropriate activation function, proper architecture, and suitable loss function help in better training performance. Parameters such as weights are updated through backward propagation mechanisms, some of which are Stochastic Gradient Descent(SGD)[123], RMSprop, or Adam[124]. The functional annotation of protein sequences can be done computationally because of their availability in a vast amount. Machine learning and deep learning are suitable for classifying the protein sequence due to the huge avail-

ability of sequential data. The information related to the protein sequence is discussed in the next section.

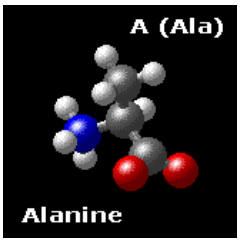
2.6.2 Primary Protein Sequence Structural Representation

Proteins are functional molecules, and the information of proteins is contained in genes. The primary structure of the protein is sequential and made of twenty essential amino acids. The study of genes is known as the genome. Genome data contains **Gene Ontology** classes and the sequence of constituent **amino acids** which form the given protein. The amino acid sequence is obtained from DNA using transcription and translation mechanisms by the cell. The naturally occurring proteins generally contain 20 essential amino acids. These are represented by one letter as shown in Table 2.5.

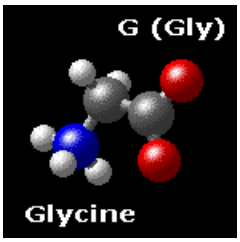
Table 2.5: Essential Amino Acids

Amino Acid	1-Letter code
Alanine	A
Cysteine	C
Aspartic Acid	D
Glutamic Acid	E
Phenylalanine	F
Glycine	G
Histidine	H
Isoleucine	I
Lysine	K
Leucine	L
Methionine	M
Asparagine	N
Proline	P
Glutamine	Q
Arginine	R
Serine	S
Threonine	T
Valine	V
Tryptophan	W
Tyrosine	Y

As shown in Table 2.5, alanine amino acid is denoted by the letter 'A,' and glycine



(a) Alanine structure



(b) Glycine structure

Figure 2.5: Structure of Alanine and Glycine amino acid

amino acid is represented by the letter 'G.' The chemical structure representation of these amino acids is shown in figure 2.5.

The structure and functions of the protein depend on the 20 amino acids and their properties. These properties of amino acids are mentioned below.

- Charge
- Hydrophobicity
- Polarity
- Aromaticity
- Presence of Hydroxyl
- Presence of Sulphur

The amino acids can be encoded mathematically into a one-hot vector of length 20. The purpose of using one-hot encoding is to retain the individuality of each representation. The presence of 6 properties can be represented by a 6 length vector where one denotes that the property is present, whereas 0 denotes that the property is absent. Therefore, for every constituent amino acid in its sequence, we have a 26 length vector. If the input sequence has length L, the input to the model is a matrix of size $26 \times L$. An input sequence can belong to multiple classes, so the annotation task of functional classes for protein sequence is a multi-label classification.

2.7 Datasets for Multi-Label Classification

The multi-label datasets cover a wide range of research domains such as music, text, image, and genomics. In this thesis, we use benchmark multi-label datasets for experimentation purposes. The deep learning approaches require the data in large amounts so that these can extract useful features from it. The categorization of multi-label datasets is shown in Figure 2.6.

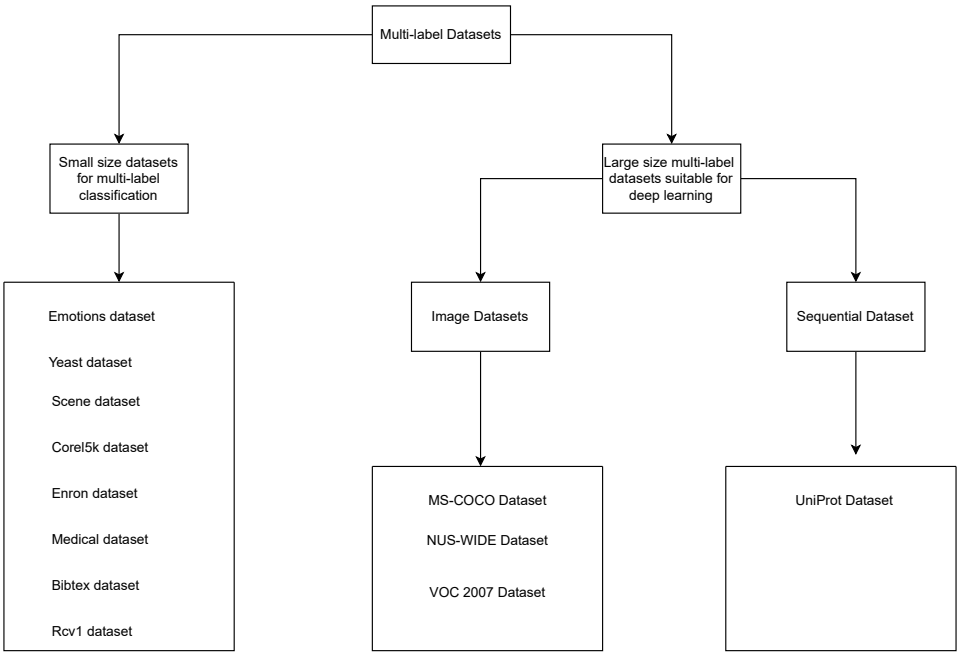


Figure 2.6: Multi-label dataset categorization

2.7.1 Multi-Label Datasets for Non-Iterative Algorithms

The non-iterative approaches use the multi-label datasets which are already in preprocessed form. The details of these benchmark datasets is given as follows.

(i) **Emotions dataset:** This is also known as a small music dataset. The emotion dataset classifies the music into amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-aggressive moods. This is based on the Tellegen-Watson-Clark model of mood. [125]

Table 2.6: Multi-label data-set description

Data	Attributes	Examples	Labels
emotions	72	593	6
yeast	103	2417	14
scene	294	2407	6
corel5k	499	5000	374
enron	1001	1702	53
medical	1449	978	45
bibtex	1836	7395	159
rcv1v2s1	47236	6000	101
rcv1v2s2	47236	6000	101
rcv1v2s3	47229	6000	101
rcv1v2s4	47235	6000	101
rcv1v2s5	47236	6000	101

(ii) **Yeast dataset:** Yeast dataset denotes the 14 possible functional classes as an output for each genome sequence represented by its instances. The yeast dataset is the genome-related dataset, and the ranking loss is in focus for optimizing for yeast-related work because of the objective of the genome data properties [3, 8]

(iii) **Scene dataset:** Each image in this dataset contains the visual numeric features in Luminance space. The scene dataset categorizes the images into six labels such as field, beach, fall foliage, sunset, field, urban, and mountain. [10]

(iv) **Corel5k dataset:** This is a benchmark dataset for annotation methods and image classification. Corel5k dataset contains five thousand images of Corel images. Corel5k is an image domain dataset that contains samples from a museum, web archive images, and newspaper images. The medical dataset contains the text corpus related to a substantial proportion of pediatric radiology activity.[126]

(v) **Enron dataset:** This dataset is generated from the email collections, and it is a subset of the enron email Corpus. These emails are categorized into 53 classes such as legal advice, humor, and strategy.[127]

(vi) **Medical dataset:** This dataset contains the data of clinical reports, which are labeled with 45 disease codes. This dataset was provided by Computational Medicine Centers. [128]

(vii) **Bibtex dataset:** This dataset is available from the ECML/PKDD 2008 discovery challenge. The bibtex dataset is annotated with the tags provided by

BibSonomy users. It has entries of BibSonomy social bookmark. [129]

(viii) **Rcv1 dataset:** These datasets are famous for text classification algorithms. Reuters has five subsets named as rcv1v2-s1,rcv1v2-s2,rcv1v2-s3,rcv1v2-s4, and rcv1v2-s5. Each subset contains 6000 articles with 101 topics. The features of articles are selected based on the methods mentioned in [130]. The summarized overview of the multi-label dataset is given in Table 2.6.

2.7.2 Multi-Label Datasets for Deep Learning Algorithms

In this section, we discuss the datasets related to deep learning, which contain a large number of instances. The details of MS-COCO, NUS-WIDE, VOC 2007, and UniProt datasets are given below.

(i) **MS-COCO dataset:** Microsoft-Common Objects in Context (MS-COCO) dataset is a widely used dataset for image classification tasks. This dataset contains the visual scenes. This dataset has been used for multi-label classification in recent times. There are 82,081 images present in the training set and 40,504 in the validation set. There are 80 classes in the MS-COCO dataset and an average of 2.9 objects per label. There are no specific labels available for multi-label learning for the test set, so multi-label approaches use the validation test for testing purposes [77]. Sample images of the MS-COCO dataset are shown in Figure 2.7.

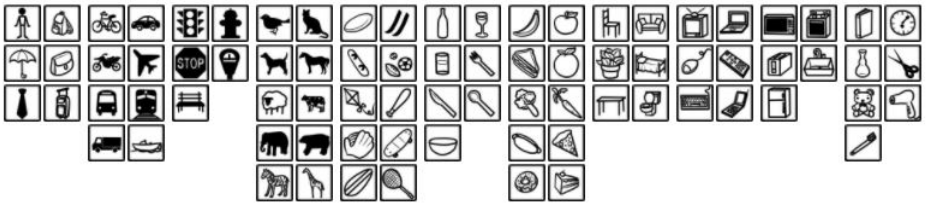


Figure 2.7: Sample images of MS-COCO dataset

(ii) **NUS-WIDE dataset:** The NUS-WIDE dataset is widely used in the multi-label image recognition task, which contains 269,648 images and 5,018 tags from Flickr. These images are further manually annotated by 81 concepts, with 2.4

Figure 2.8: Sample images of NUS-WIDE dataset

concept labels per image on average. Following its official settings, we use 161,789 images for training and 107,859 images for the test set. This dataset provides four different size images: large size, middle size, small size, and original size [131]. In this thesis work, we use a small size for our experimental purpose. Sample images of the NUS-WIDE dataset are shown in Figure 2.8.

(iii) **UniProt dataset:** The UniProt dataset is accessible from the http://uniprot.org webpage, and its SwissProt subset contains more than five lacks sequences having Gene Ontology IDs. In this thesis work, the SwissProt subset of the UniProt database[132] is used after being acquired from http://uniprot.org as starting point (using the query"goa:(*) AND reviewed: yes") at the date of download on 5 August 2019. The raw data acquired from UniProt is shown in Table 2.7.

Table 2.7: Raw data acquired from UniProt

Sequence Samples	535119
Label Samples	535119
Max Sequence Length	35213
Min Sequence Length	2
Max Label Length	258
Min Label Length	1
Total Labels	2970815
Total Unique Labels	28234

Figure 2.9: Sample images from VOC 2007 dataset

Clearly, the amount of data is large enough to become a challenge for any kind of deep neural network.

(iv) **VOC 2007 dataset:** The PASCAL VOC 2007 dataset is the collection of consumer photographs taken from the photo-sharing website Flickr. This dataset is well used for multi-label classification tasks. There are 9963 images from the 20 classes in the VOC dataset [76]. These classes cover person, animal, vehicle, and indoor-related images. Sample images of the VOC 2007 dataset are shown in Figure 2.9.

2.8 Performance Evaluation Metrics

Multi-label classification is considered the generalized version of the multi-class classification. The metrics used to evaluate the performance of multi-label classification are different from the multi-class classification. These evaluation metrics are discussed in this section. In this thesis, we categorized the evaluation metrics into two categories. In the first category, evaluation metrics related to non-iterative algorithms are discussed, and in the second category, evaluation metrics related to deep learning algorithms are discussed.

2.8.1 Evaluation Metrics for Non-Iterative Multi-Label Algorithms

In this thesis work, hamming loss, ranking loss, one error, coverage, and, average precision metrics are used for the performance evaluation of non-iterative approaches. For a given test set $\{(x_1, Y_1), (x_2, Y_2), ..., (x_{N'}, Y_{N'})\}$ having unseen instances these metrics are evaluated. Let x_i denotes the i^{th} instance and Y_i denotes the label set of the i^{th} instance. For the total number of $N^{'}$ samples of test set the details of evaluation metrics are mentioned as follows.

(i) **Hamming Loss:** Misclassification of instance labels is evaluated using hamming loss. This misclassification can be represented by XOR logic and denoted by the following equation.

$$\text{hamming loss} = \frac{1}{N'} \sum_{i=1}^{N'} \frac{1}{Q} |h(x_i) \triangle Y_i|, \tag{2.8}$$

where Q is the total number of possible class labels and $\triangle$ stands for the symmetric difference between predicted and target labels. The small value of hamming loss is considered better than the large value.

(ii) **Ranking Loss:** The average fraction of label pairs, arranged in reverse order, is evaluated by the Ranking Loss.

$$\text{ranking loss} = \frac{1}{N'} \sum_{i=1}^{N'} \frac{1}{|Y_i||\overline{Y_i}|} \tag{2.9}$$
$$|\{(y_1, y_2)| f(x_i, y_1) \leq f(x_i, y_2), (y_1, y_2) \in Y_i \times \bar{Y_i}\}|$$

The $\bar{Y}$ denotes the complementary set of Y in label space $\mathcal{Y}$. The mathematical expression of ranking loss evaluates the average fraction of label pairs that are reversely ordered for an input instance. The performance is perfect when the ranking loss is zero. The smaller value of ranking loss denotes better performance.

(iii) **One error:** One error evaluates the missing of the top-ranked label associated

with the input instance. One error can be described as follows:

$$\text{one-error} = \frac{1}{N'}\sum_{i=1}^{N'}\left[\left[argmax_{y\in\mathcal{Y}} f(x_i, y)\right] \notin Y_i\right] \tag{2.10}$$

The small value of one error is considered better than the large value.

(iv) **Coverage:** This metric evaluates the measure to reach all the labels associated with the input instance. It can be related to the precision value at the perfect recall.

$$\text{coverage} = \frac{1}{N'}\sum_{i=1}^{N'} max_{y\in Y_i} rank_f(x_i, y) - 1. \tag{2.11}$$

As discussed in Section 2.1 $rank_f$ is derived from the real-valued function $f(.,.)$, which maps the outputs of $f(x_i, y)$ for any $y \in \mathcal{Y}$ to $\{1, 2, ..., Q\}$ such that if $f(x_i, y_1) > f(x_i, y_2)$ then $rank_f(x_i, y_1) < rank_f(x_i, y_2)$. The smaller value of coverage denotes better performance.

(v) **Average Precision:** This evaluation metric measures the mean fraction of output labels, which are ranked higher than a particularly associated output label $y \in Y$.

$$\begin{aligned}&\text{average precision} = \\ &\tfrac{1}{N'}\textstyle\sum_{i=1}^{N'}\tfrac{1}{|Y_i|}\sum_{y\in Y_i}\tfrac{|\{y' | rank_f(x_i,y')\leq rank_f(x_i,y), y'\in Y_i\}|}{rank_f(x_i,y)}\end{aligned} \tag{2.12}$$

The large value of average precision is considered better than the small value.

2.8.2 Evaluation Metrics For Deep Learning based Multi-Label Algorithms

The complete details of evaluation metrics used for deep learning based algorithms is given in this section. The confusion matric is shown in Table 2.8.

In Table 2.8 the TP values are the correctly predicted positive values, and TN values are the correctly predicted negative values. The FP values are the wrong predicted positive values, and FN values are the wrong predicted negative values.

Table 2.8: Multi-label data-set description

Total population = Positive + Negative		**Predicted**	
		Positive	**Negative**
Ground Truth	Positive	True Positive (TP)	False Negative(FN)
	Negative	False Positive (FP)	True Negative (TN)

(i) **Accuracy:** Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observations to the total observations. Accuracy is a good measure, but it is only preferred in those cases when datasets are symmetric where values of false positives and false negatives are almost the same. The following mathematical expression denotes the formulation of accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.13}$$

(ii) **Precision:**Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Following mathematical expression denotes the formulation of precision.

$$Precision = \frac{TP}{TP + FP} \tag{2.14}$$

(iii) **Recall:**Recall is the ratio of correctly predicted positive observations to the all observations in actual class. Following mathematical expression denotes the formulation of recall.

$$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

(iv) **F1 score:** F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. The F1 score is usually more useful than accuracy, especially for the case when class distribution is uneven. Following mathematical expression denotes the formulation of F1 Score.

$$\text{F1 score} = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{2.16}$$

In this thesis, we have used the hamming loss, ranking loss, one error, coverage,

and average precision evaluation metrics to evaluate the non-iterative approaches. The precision, recall, and F1 score are used to evaluate the deep learning based multi-label algorithms in the image domain. The sequential data domain multi-label algorithms are evaluated based on precision, recall, and subset accuracy. These evaluation metrics are used for a fair comparison with existing state-of-the-art approaches in the respective domains.

2.9 Statistical Tests

Statistical tests provide information about the significance of performance results when the proposed approach is compared to the existing approaches. We use the following test and studies to test the performance of the proposed approaches.

2.9.1 Wilicoxon Signed Ranksum Test

Wilcoxon test is a non-parametric test to compare the data. It is preferred when the data pair is non-normally distributed. In the thesis, we use this test to find the significance of the performance of the algorithm for particular evaluation metrics [133],[134].

2.9.2 Ablation Study

The ablation study is useful for analyzing the effect of a specific component of the algorithm. In this thesis, we use the ablation study to see the effect of the number of layers in GCN. The impact of layers in GCN is analyzed by using the ablation study. The effect of pairnorm is shown by the ablation study for the proposed GCN based multi-label classifiers.

Chapter 3

Broad Learning based Multi-Label Classification

In this chapter, we have proposed multi-label classifiers based on Broad Learning System. The Broad Learning System (BLS) is a non-iterative approach to solve multi-class problems. We have adapted BLS to solve multi-label problems and named it a Multi-Label Broad Learning System (ML-BLS). The ML-BLS is a non-iterative multi-label classifier that contains less number of parameter and hyperparameter in comparison to the iterative algorithms, which contains gradient descent to optimize the loss function. The ML-BLS is a neural network architecture. The neural networks based approaches have the shortcoming that these algorithms work like the black box because these approaches lack the ability to explain the results and neural networks based approaches have the inability to reveal enough information about the system it approximates. The fuzzy system possesses better interpretability of results. By considering the benefit of fuzzy systems for interpretability, we have also proposed a neuro-fuzzy approach named a multi-label fuzzy broad learning system (ML-FBLS) which improves the performance of ML-BLS. These non-iterative multi-label classifiers use the pseudoinverse to compute the parameters of the classifiers. These ML-BLS and ML-FBLS algorithms reduce the training time along with optimization of the evaluation measures of multi-label classification. The details of the proposed algorithms are discussed in further sections.

3.1 Introduction

The neural network based architectures are widely applied to solve various kinds of classification and regression problems. The neural networks have the universal approximation capability to solve the research problems. Traditional neural networks use gradient-descent algorithms for the optimization of network parameters. Data dimensions also increase nowadays with the size of data. The algorithm can find it difficult to sustain its accessibility due to the direct feeding of high dimensional raw data [29]. To resolve this issue, feature extraction and dimension reduction are used, which help the network to sustain its accessibility. Recently a neural network named the BLS [29, 135] has been proposed, which is based on non-iterative learning for the optimization of the parameters. BLS is able to handle the high volume data without the need for deep learning architecture [29, 136]. The basic idea of BLS is to use the non-iterative procedure to find the parameters of the neural network and map the input data to features. BLS takes the mapped features as input to the neural network. The broad learning system has been used in different areas of research in recent times. The facial expression recognition task has been done by the BLS to provide robust real-time performance [137]. The BLS is capable of universally approximating the functions because of its neural network structure. As discussed earlier, The neural network based algorithms are unable to interpretrate the results and unable to explain the reasons of its performance. Fuzzy rules and fuzzy systems are able to better interpretrate the results. Hence neuro fuzzy hybrid approaches provides the benefits of both neural network and fuzzy system based algorithms[69].

BLS is used to create a hybrid classification system with a fuzzy subsystem model to take the benefits of both neural networks and fuzzy subsystems [69, 136]. The BLS is useful to solve various problems of different domains because of its fast learning capability and ability to handle large-scale data. BLS provides a learning solution to classify the multi-class classification problem. BLS is not able to separate the labels of those input samples which have multi-label properties. To make the BLS classify the multi-label data, we propose two BLS based multi-label classifiers named ML-BLS

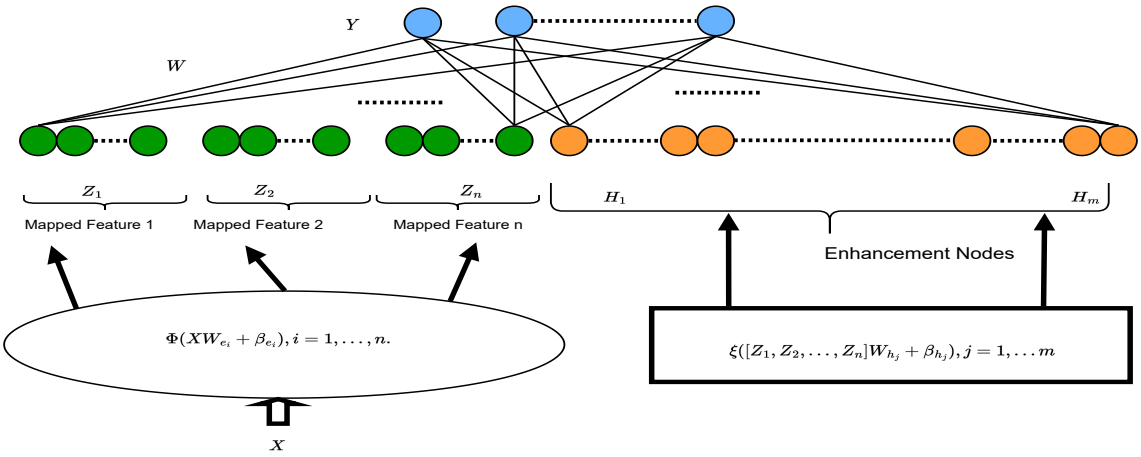


Figure 3.1: Broad Learning System

and ML-FBLS in this chapter. We have discussed the details of ML-BLS in the next section.

3.2 Multi-Label Classifier based on Broad Learning System: ML-BLS

The ML-BLS is an algorithm adaptation based adaptation of BLS for multi-label classification. In BLS, initially, inputs are mapped to construct a set of mapped features. Then enhancement nodes are created with the help of mapped features. Suppose input data is presented by X and it is projected to become the mapped feature Z_i, using $\Phi_i(XW_{e_i} + \beta_{e_i})$, where Z_i, is i^{th} mapped feature, W_{e_i} represents the random weights with proper dimensions. Set of first i concatenated groups of mapped features represented as $Z^i \equiv [Z_1, ..., Z_i]$, in broad learning system. Similarly, H_j, denotes j^{th} group of enhancement nodes calculated using, $\xi_j(Z^n W_{h_j} + \beta_{h_j})$ and set of the first j concatenated groups of enhancement nodes are represented as $H^j \equiv [H_1, ..., H_j]$, in broad learning system. Let, input data denoted by X, contains N samples and each of these samples has M dimensions. The raw output of BLS is denoted as $Y \in R^{N \times Q}$. For n feature mappings, we can represent each mapping in the mathematical equation as :

$$Z_i = \Phi(XW_{e_i} + \beta_{e_i}), i = 1, ..., n. \tag{3.1}$$

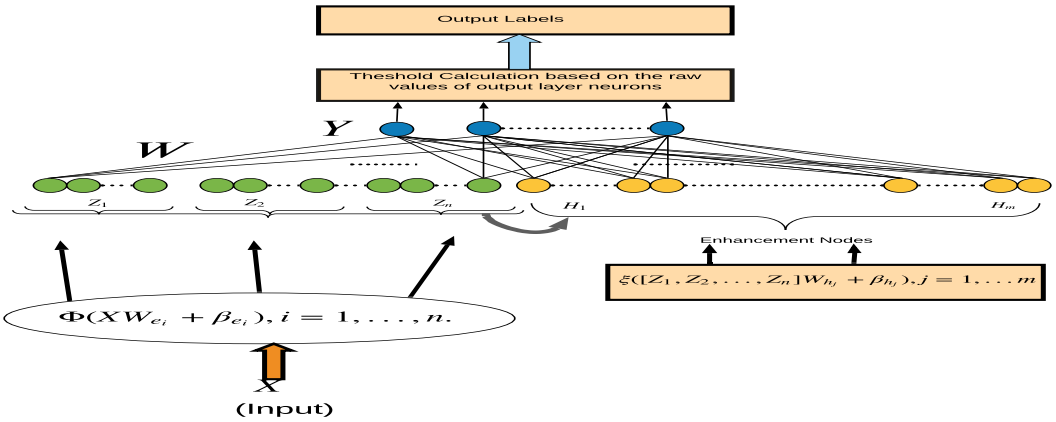


Figure 3.2: Multi-Label Broad Learning System Neural Network

where W_{e_i} and β_{e_i} are generated randomly.

The set of all the feature nodes are denoted as $Z^i \equiv [Z_1, ..., Z_i]$, the m^{th} group of enhancement nodes is denoted as

$$H_m \equiv \xi(Z^n W_{h_m} + \beta_{h_m}) \tag{3.2}$$

The nonlinear functions are used to model the target variables (class, labels, or scores) that vary non-linearly with their explanatory variables. Nonlinear means that the output cannot be reproduced from a linear combination of the inputs. The neural networks without nonlinear functions behave just like single-layer perceptron because summing these layers would give you just another linear function. The ξ and ϕ denote the nonlinear functions in the broad learning system.

Hence, the following equation represents the broad learning system.

$$\begin{aligned} Y &= [Z_1, ..., Z_n | \xi(Z^n W_{h_1} + \beta_{h_1}), ..., \xi(Z^n W_{h_m} + \beta_{h_m})]W \\ &= [Z_1, ..., Z_n | H_1, ..., H_m]W \\ &= [Z^n | H^m]W \end{aligned} \tag{3.3}$$

where $W = [Z^n|H^m]^+ Y$ denotes the connecting weights of the broad learning system. Pseudoinverse using ridge regression is used to compute $W \in \mathcal{R}^{(m+n)\times Q}$. We can

represent the raw output of BLS as follows

$$f(x) = [Z^n|H^m]W \tag{3.4}$$

The above output is provided to compute the threshold computation to create ML-BLS for multi-label classification which is discussed in section 3.4.

3.3 Multi-Label Classifier based on Fuzzy Broad Learning System: ML-FBLS

In this section, we present the fuzzy system based broad learning system for multi-label classification. The fuzzy broad learning system (FBLS) is a novel neuro-fuzzy algorithm for multi-class classification and regression problems [69]. It creates groups of different fuzzy subsystems to process the input data. The values produced by fuzzy-subsystems are used for nonlinear transformations using the enhancement layer. The enhancement nodes in the enhancement layer are created by the output of each fuzzy subsystem. This preserves the input characteristic by the nonlinear transformation. The output of the enhancement layer and defuzzification outputs of all fuzzy subsystems are concatenated and provided to the output layer as an input, as shown in Figure 3.3. Let there are n fuzzy subsystems, m enhancement groups in the FBLS. The input data matrix of size $N \times M$ is represented as $X = (x_1, x_2, ..., x_N)^T \in \mathbb{R}^{N \times M}$ where x_s=$(x_{s1}, x_{s2}, ..., x_{sM})$, $s = 1, 2, ..., N$. The fuzzy rules in the fuzzy subsystem can be denoted by

If x_{s1} is A^i_{k1} and x_{s2} is A^i_{k2}... and x_{sM} is A^i_{kM}
then $z^i_{sk} = f^i_k(x_{s1}, x_{s2}, ..., x_{sM}), k = 1, 2..., K_i$
where K_i is the fuzzy rule in the i^{th} fuzzy subsystem.

There are two main types of fuzzy systems named as Mamdani fuzzy system [138] and the Takagi-Sugeno (TS) fuzzy system [139]. TS fuzzy system is suitable for FBLS because every fuzzy rule in TS fuzzy system can be represented as the function of the input, and its subsystems can be used as nodes in a fuzzy broad learning system.

Let the fuzzy subsystem using the TS fuzzy system is denoted by

$$z_{sk}^{i} = f_{k}^{i}(x_{s1}, x_{s2}, ..., x_{sM}) = \Sigma_{t=1}^{M} \alpha_{ki}^{i} x_{st} \tag{3.5}$$

where α_{ki}^{i} represents the coefficient. The gaussian membership function corresponding to fuzzy set A_{kt}^{i} with center c_{kt}^{i} and width σ_{kt}^{i} is defined as

$$\mu_{kt}^{i}(x) = e^{-\left(\frac{x-c_{kt}^{i}}{\sigma_{kt}^{i}}\right)^{2}} \tag{3.6}$$

The τ_{sk}^{i} denotes the fire strength of the k^{th} fuzzy rule in the i^{th} fuzzy subsystem which is denoted as below

$$\tau_{sk}^{i} = \prod_{t=1}^{M} \mu_{kt}^{i}\left(x_{st}\right) \tag{3.7}$$

The following equation represents the weighted fire strength ω_{sk}^{i} related to k^{th} fuzzy rule in i^{th} fuzzy subsystem

$$\omega_{sk}^{i} = \frac{\tau_{sk}^{i}}{\Sigma_{k=1}^{K_i} \tau_{sk}^{i}} \tag{3.8}$$

where τ_{sk}^{i} is fire strength.

The s^{th} training example x_s is passed to i^{th} fuzzy subsystem. The output vector Z_{si}, generated by the i^{th} fuzzy subsystem for s^{th} training sample x_s is denoted as

$$Z_{si} = (\omega_{s1}^{i} z_{s1}^{i}, \omega_{s2}^{i} z_{s2}^{i}, ..., \omega_{sK_i}^{i} z_{sK_i}^{i}) \tag{3.9}$$

For all training samples X, the output of i^{th} fuzzy subsystem can be denoted as

$$Z_{i} = (Z_{1i}, Z_{2i}, ..., Z_{Ni})^{T} \in \mathbb{R}^{N \times K_i}, i = 1, 2, ..., n \tag{3.10}$$

The intermediate outputs generated by the fuzzy subsystems are used to create the enhancement node groups, and the intermediate output matrix of n fuzzy subsystems is denoted by follows:

$$Z^{n} = (Z_{1}, Z_{2}, ..., Z_{n}) \in \mathbb{R}^{N \times (K_1 + K_2 + ... + K_n)} \tag{3.11}$$

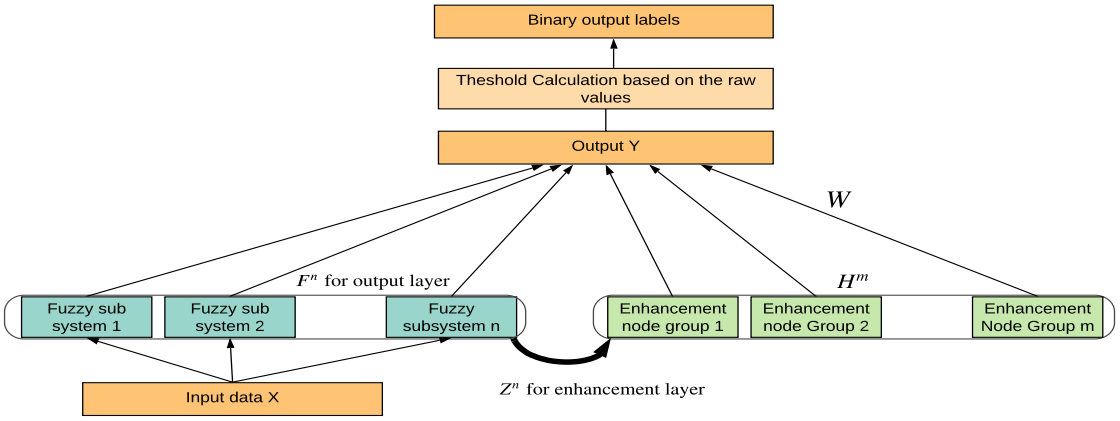


Figure 3.3: Multi-Label Fuzzy broad learning system

Then Z^n is used to create the enhancement nodes for nonlinear transformation. The output of enhancement node groups can be denoted by

$$H^m = (H_1, H_2, ..., H_m) \in \mathbb{R}^{N\times(L_1+L_2+...+L_m)} \tag{3.12}$$

where L_j denotes the neurons in the j^{th} enhancement node group, $H_j = \xi_j(Z^n W_{h_j} + \beta_{h_j}) \in \mathbb{R}^{N\times L_j}$ is the output of j^{th} enhancement group, W_{h_j}, and β_{h_j} are the weight and bias randomly generated from the range of $[0, 1]$ and j has the values between 1 to m. The input for output layer Y is the defuzzification output generated by each fuzzy subsystem concatenated with H^m. The output of output layer Y has Q labels, and then each fuzzy subsystem should be represented as multioutput. Hence the output of i^{th} fuzzy subsystem is represented as a vector for input sample x_s as follows:

$$\begin{aligned} F_{si} &= (\Sigma_{k=1}^{K_i}\omega_{sk}^i(\Sigma_{t=1}^{M}\delta_{k1}^i\alpha_{kt}^i x_{st}), ..., \Sigma_{k=1}^{K_i}\omega_{sk}^i(\Sigma_{t=1}^{M}\delta_{kQ}^i\alpha_{kt}^i x_{st})) \\ &= \Sigma_{t=1}^{M}\alpha_{kt}^i x_{st}(\omega_{s1}^i, ..., \omega_{sK_i}^i) \begin{pmatrix} \delta_{11}^i & \cdots & \delta_{1Q}^i \\ \vdots & \ddots & \vdots \\ \delta_{K_i1}^i & \cdots & \delta_{K_iQ}^i \end{pmatrix} \end{aligned} \tag{3.13}$$

where $\delta_{kq}^i\alpha_{kt}^i(q = 1, 2, ..., Q)$ representes the initial cofficient parameter α_{kt}^i and δ_{kQ}^i is introduced to make the matrix form of the consequent part of each fuzzy rule. The output for all training instances produced by i^{th} fuzzy subsystem is represented as

follows:

$$F_i = (F_{1i}, F_{2i}, ..., F_{Ni})^T \triangleq D\Omega^i\delta^i \in \mathbb{R}^{N\times Q} \tag{3.14}$$

where $D = diag\Sigma_{t=1}^{M}\alpha_{kt}^i x_{1t}, ..., \Sigma_{t=1}^{M}\alpha_{kt}^i x_{Nt}$, and
$\Omega^i = \begin{pmatrix} \omega_{11}^i & \cdots & \omega_{1K_i}^i \\ \vdots & \ddots & \vdots \\ \omega_{N1}^i & \cdots & \omega_{NK_i}^i \end{pmatrix}$ and $\delta^i = \begin{pmatrix} \delta_{11}^i & \cdots & \delta_{1Q}^i \\ \vdots & \ddots & \vdots \\ \delta_{K_i1}^i & \cdots & \delta_{K_iQ}^i \end{pmatrix}$.

Aggregate output of n fuzzy subsystems of the output layer Y is denoted as:

$$\begin{aligned} F^n &= \Sigma_{i=1}^{n} F_i = \Sigma_{i=1}^{n} D\Omega^i\delta^i = D\begin{pmatrix}\Omega^1 & \cdots & \Omega^n\end{pmatrix}\begin{pmatrix}\delta^1 \\ \vdots \\ \delta^n\end{pmatrix} \\ &\triangleq D\Omega\Delta \in \mathbb{R}^{N\times Q} \end{aligned} \tag{3.15}$$

where $\Omega = (\Omega^1, ..., \Omega^n) \in \mathbb{R}^{N\times(K_1+K_2+,...,+K_n)}$ is the matrix which denotes the fire strength ω_{sk}^i and $\Delta = ((\delta^1)^T, ..., (\delta^n)^T)^T \in \mathbb{R}^{(K_1+K_2+,...,K_n)\times Q}$

Output F^n of all the fuzzy subsystem together with H^m is sent to the output layer Y of Fuzzy BLS. The weight matrix connecting the enhancement layer to the output layer Y is denoted as $W_e \in \mathbb{R}^{(L_1+L_2+,...,+L_m)\times Q}$, while the weights connecting the fuzzy subsystems to top layer are all set to be 1. Therefore, the final output of FBLS is represented as follows:

$$\begin{aligned} \hat{Y} &= F^n + H^m W_e \\ &= D\Omega\Delta + H^m W_e \\ &= (D\Omega, H^m)\begin{pmatrix}\Delta \\ W_e\end{pmatrix} \end{aligned} \tag{3.16}$$

where W_e denotes the weights connecting the Y and enhancement layer. F^n represents the output of all fuzzy subsystem.

The Eq.(3.16) can be further written as follows in the proper form of linear equation:

$$\hat{Y} \triangleq (D\Omega, H^m)W \tag{3.17}$$

where W is the parameter matrix of a fuzzy BLS consisting of Δ and W_e.

Now FBLS can be written in the mathematical form as follows:

$$Y = (D\Omega, H^m)W \tag{3.18}$$

the parameter W of above equation can be computed using pseudoinverse as represented follows:

$$W = (D\Omega, H^m)^+ Y \tag{3.19}$$

where $(D\Omega, H^m)^+ = ((D\Omega, H^m)^T (D\Omega, H^m))^{-1} (D\Omega, H^m)^T$

Above Eq.(3.19) represents the computation of output layer weights using the pseudoinverse. Using the pseudoinverse is a noniterative technique that makes the training faster than the iterative procedures. The raw outputs at layer Y can be represented as follows

$$f(x) = \hat{Y} = (D\Omega, H^m)W \tag{3.20}$$

The values of $f(x)$ are provided to the threshold function to create ML-FBLS which is discussed in the next section.

3.4 Adaptative Thredhold for Multi-Label Classification

In this section, we provide the details of the multi-label adaptation of the proposed randomized non-iterative algorithms ML-BLS and ML-FBLS. These approaches use the fast learning advantage of non-iterative algorithms. All of these approaches can be considered algorithm adaptation approaches. For all the proposed multi-label classifiers, the output layer Y represents the numerical raw outputs $f(x)$ corresponding to the particular class or label. We have used these numerical outputs as an input in the calculation of the threshold. A threshold function is needed to map these numerical values into the associated set of labels. Usually fixed threshold, zero constant function based threshold are used as threshold function for multi-label classification [25, 57]. There are multiple thresholding methods that have also been proposed, such

as probability-based threshold, the margin between irrelevant and relevant labels based on constant threshold [8, 57, 25].

As ML-BLS and ML-FBLS are neural networks, these networks provide the numerical values on the output layer Y. These numerical values are used for the ranking of labels. Taking into consideration of ranking of labels, we adapt the threshold function [8] with ML-BLS and ML-FBLS for learning the multi-label problems. This threshold function finds the boundary between irrelevant and relevant labels. This boundary is related to the whole training set, so there is a need to use the threshold function, which should be dynamic, and adaptive to the whole data set. The threshold function is expressed as a linear function as follows:

$$t(x) = w^T.c(x) + b, \tag{3.21}$$

where $c(x) = (c_1(x), c_2(x), ..., c_Q(x))$ is a Q-dimensional vector. The $c(x)$ represents the output vector of $f(x)$ after processing it using activation function. This threshold function is linear due to the requirement of a few parameters by a linear function and its simple implementation.For each training instance (x_i, y_i) $(1 \leq i \leq N)$, and output vector at $c(x_i) = (c_1^i, c_2^i, ..., c_Q^i)$ and the target values are set as

$$t(x_i) = argmin_t(|k|k \in Y_i, c_k^i \leq t| + |l|l \in \overline{Y_i}, c_l^i \geq t|) \tag{3.22}$$

The matrix equation $\phi.w^{'} = t$ can be used to learn the parameter of Eq.(3.21). The dimension of matrix ϕ are $N \times (Q+1)$, dimension of vector $w^{'}$ and t are $(Q+1)$ and N respectively. The t is a N dimensional vector corresponding to training set. To obtain the optimal parameter w and b of Eq.(3.21), the least square method is used on the basis of solving t as mentioned below.

$$min_{w,b} \sum_{i=1}^{N} ((w^T.c(x_i)) + b - t(x_i))^2 \tag{3.23}$$

The distance between $(w^T.c(x_i)) + b$ and t_{x_i} is minimized by Eq.(3.23). The actual raw outputs are used for the ranking of labels. The threshold function creates the threshold

using this adaptive threshold. In this threshold function, the minimum value of t is computed, which minimize the RHS of Eq. 3.22. The Eq. 3.23 is used to compute the value of parameters w and b of Eq. 3.21. This equation minimizes the squared sum of the distances between $w^t.c(x_i)$ and $t(x_i)$ in the training set.

After the training we have w and b for the threshold function mentioned in Eq.(3.21) for training set. At the testing time, unseen instance x is fed to the trained ML-BLS, and ML-FBLS networks and the output vector $c(x)$ is computed at the output layer by using following mathematical expression.

$$\text{Final labels corresponds to } Y = \begin{cases} 1, & c_k(x_i) \geq t(k) \\ -1, & c_k(x_i) < t(k) \end{cases} \tag{3.24}$$

where, $i = 1, ..., N^{'}$ and $k = 1, ..., Q$. The complete algorithms for multi-label classification using ML-BLS, and ML-FBLS are described in Algorithms 1, 2 respectively.

3.5 Experimentation and Results

In this section, the experimental results related to proposed ML-BLS and ML-FBLS are discussed to verify the effectiveness of these proposed multi-label non-iterative approaches. Multi-label classification experimentation is carried out on 12 benchmarks multi-label datasets[1] [2], which are described in Table 2.6. Comparative results based on evaluation metrics are shown in Table 3.2 to Table 3.13. All the experiments related to the comparison of multi-label classification are performed on MATLAB 2017a. Five-fold cross-validation is carried out on all 12 multi-label datasets. For each dataset, we have used five-fold cross-validation. In each fold, 1 set is used for testing, and the rest four sets are used as the training set. For this purpose, we have generated the indexing of instances for each multi-label dataset for each fold. The indexing for each fold is the same for all the compared approaches. Using the five-fold cross-validation, 80% of total data is used as training data and 20% for tests for all

[1]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2]http://mulan.sourceforge.net/datasets-mlc.html

Algorithm 1: Multi-Label Broad Learning System

Input : Training samples $(X, Y) \in R^{(N\times(M+Q))}$

Output: A ML-BLS with output weight W, threshold parameters w and b

for `i=1;i<=n` **do**
  Generate W_{e_i},β_{e_i} randomly;
  Calculate $Z_i = \Phi(XW_{e_i} + \beta_{e_i})$;
end
Create a set for the group of feature nodes $Z^n = [Z_1, ..., Z_n]$ using Eq.(3.1);
for `j=1;i<=m` **do**
  Generate W_{h_j},β_{h_j} randomly;
  Calculate $H_j = \xi(Z^n W_{h_j} + \beta_{h_j})$
end
Create a set for the group of enhancement nodes using Eq.(3.2). $H^m = [H_1, ..., H_m]$;
Concatenate the mapped feature and enhancement nodes denoted as $[Z^n|H^m]$
Calculate the weight matrix W using $W = [Z^n|H^m]^+Y$;
Using weight matrix W, Compute raw values $f(x)$ at output layer Y for input sample x
Compute threshold function parameters w^T and b using Eq.(3.22) and Eq.(3.23) as follows. $min_{w,b} \sum_{i=1}^{n}((w^T.c(x_i)) + b - t_{x_i})^2$.
Calculate outputs $c(x)$ and threshold for each test data instance according to Eq.(3.22) and Eq.(3.23)
Compare $c(x)$ with a threshold value and obtain multi-label identification final outputs according to Eq.(3.24)

the datasets.

The root means square error (RMSE) is minimized between the actual labels and predicted labels, and the grid search is used for hyperparameter and parameter tuning. The adaptive threshold is used to find out the final labels for multi-label classification. The five metrics are evaluated after computing the hyperparameters, parameters, and weights. The same tuning procedure is applied to compare the multi-label approaches for 12 datasets. The performance of ML-BLS, and ML-FBLS is compared with algorithm adaptation based multi-label classification algorithms RankSVM, ML-ELM, BP-MLL, and ML-KRR/ML-KELM [8, 3, 57, 24].

Algorithm 2: Multi-Label Fuzzy Broad Learning System

Input : Training samples $(X, Y) \in R^{(N\times(M+Q))}$, enhancement nodes L_j, numbers of fuzzy rules K_i, fuzzy subsystems n, and enhancement node groups m

Output: A ML-FBLS with output weight W, threshold parameters w and b

Initialization of α_{kt}^i in f_k^i function by uniform distribution in the range of $[0, 1]$
for *i=1;i<=n* **do**
    Obtain clustering centers k_i for training data X using k-means algorithm
    Use K_i clustering centers to initialize centers of membership function
    for *s=1;s<=N* **do**
        compute Z_{si} using Eq.(3.9)
        compute F_{si} using Eq.(3.13)
    end
    Compute Z_i using Eq.(3.10)
    Compute F_i using Eq.(3.14);
end
Compute Z^n using Eq.(3.11)
Compute H^m using Eq.(3.12);
Compute F^n using Eq.(3.15)
Compute W using Eq.(3.19);
Using weight matrix W, Compute raw values $f(x)$ at output layer Y for input sample x
Compute threshold function parameters w and b using Eq.(3.22)and Eq.(3.23) as follows. $min_{w,b} \sum_{i=1}^{n}((w^T.c(x_i)) + b - t_{x_i})^2$.
Calculate outputs $c(x)$ and threshold for each test data instance according to Eq.(3.22) and Eq.(3.23)
Compare $c(x)$ with a threshold value and obtain multi-label identification final outputs according to Eq.(3.24)

3.5.1 Experimental Setup

The experimental results provided by the RankSVM approach are based on the details mentioned in [8]. For the RankSVM approach, the regularization parameter is chosen as 1, the tolerance level for the regularization parameter is chosen as 10^{-5}. The number of iterations is chosen 50 for the RankSVM. The regularization parameter 2^0 is used for emotions, yeast, and scene datasets. For corel5k, enron, medical, bibtex, rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5 datasets regularization parameter 2^1 is used. In the implementation details of RankSVM, the regularization parameter is mentioned as the cost parameter. The training procedure of RankSVM

is slow, and in the experimentation, it heavily depends on the number of instances of data and the output number of labels also. For the BP-MLL approach, the number of nodes in the hidden layers is chosen as 20% of the number of input features. The learning rate is chosen with the value of 0.05. The number of epochs is selected as 100 for the emotions, yeast, scene, enron, and medical dataset. The number of epochs is 50 for corel5k, bibtex, rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5 datasets, as this is mentioned as the default value in the experimental settings of BP-MLL.

For the ML-ELM approach, the number of nodes in the hidden layer is 60, 100, 85, and 250 for emotions, yeast, scene, corel5k datasets for the ML-ELM approach. For the datasets, enron, medical, bibtex, rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5, the number of nodes in the hidden layer used 120 for the ML-ELM approach. ML-ELM approach is based on the details mentioned in [57] for multi-label classification using the basic ELM. For ML-KRR/ML-KELM, the kernel parameter 2^{-2} and cost parameter 2^0 are used for emotions, yeast, and scene datasets. For corel5k, enron, medical, bibtex, rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5 datasets kernel parameter 2^{-2} and cost parameter 2^1 are used. The cost parameter C for ML-ELM and ML-KRR/ML-KELM is considered as the inverse of the regularization parameter λ mentioned in the RVFL literature.

Table 3.1: Optimal parameter settings for multi-label classifiers

	BP-MLL[3]	ML-ELM[57]		ML-KRR/ ML-KELM [24]		ML-BLS			ML-FBLS		
	Hidden layer nodes	Hidden layer nodes	C=$\frac{1}{\lambda}$	σ	C=$\frac{1}{\lambda}$	N_f	N_m	N_e	N_r	N_t	N_e
Bibtex	367	120	0.5	2^{-2}	2^1	51	47	44	10	3	2
Corel5k	100	250	0.5	2^{-2}	2^1	10	14	55	1	12	19
Emotions	15	60	0.5	2^{-2}	2^0	51	10	13	88	5	102
Enron	200	120	0.5	2^{-2}	2^0	36	26	28	67	3	19
Scene	59	85	0.5	2^{-2}	2^0	25	18	37	23	97	79
Yeast	21	100	0.5	2^{-2}	2^0	10	10	35	89	39	93
Medical	290	120	0.5	2^{-2}	2^1	24	7	58	23	97	79
rcv1v2-s1	9448	120	1	2^{-2}	2^1	64	66	63	44	33	35
rcv1v2-s2	9448	120	1	2^{-2}	2^1	39	55	35	39	7	35
rcv1v2-s3	9448	120	1	2^{-2}	2^1	59	6	16	41	26	35
rcv1v2-s4	9448	120	1	2^{-2}	2^1	29	35	53	50	9	21
rcv1v2-s5	9448	120	1	2^{-2}	2^1	34	13	33	40	35	33

The ML-FBLS mapping is similar to the functional mapping used in [54, 68]. The

ML-BLS is the adaptation of BLS for multi-label classification. The hyperparameters of ML-BLS consist of the numbers of feature nodes N_f , mapping groups N_m, and enhancement nodes N_e and these are mentioned in Table 3.1. The hyperparameters of numbers of rules N_r in each fuzzy subsystem, Numbers of fuzzy subsystems N_t, and the number of enhancement nodes N_e for ML-FBLS are mentioned in Table 3.1. We perform grid search for the hyperparameters of ML-FBLS N_r, N_t, and N_e in the range of $[1, 100]$, $[1, 100]$, and $[1, 100]$ respectively. For ML-BLS the range of hyperparameters N_f , N_m, and N_e is set in the range of $[1, 100]$,$[1, 100]$, and $[1, 150]$ respectively. The optimized parameters for the compared approaches are mentioned in Table 3.1. The activation function used for ML-BLS and ML-FBLS are sigmoid and tanh respectively which are referred from the BLS based papers.

3.5.2 Results and Performance Evaluation

In this section, we provide the experimental results performed on the twelve multi-label datasets from various domains as mentioned in 2.6. The optimum performance is highlighted in bold for each evaluation metric in table 3.2 to 3.13. further, the average performance on twelve datasets for five evaluation metrics is described in table 3.14.

Table 3.2: Comparison results performed on corel5k dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0109±0.0001	0.147±0.0034	0.776±0.0184	0.323±0.0056	0.209±0.0085
ML-ELM[57]	0.0104±0.0001	0.561±0.0154	**0.377±0.0081**	0.875±0.0138	**0.254±0.0134**
BP-MLL[3]	0.0095±0	0.273±0.0109	0.931±0.0129	**0.263±0.0053**	0.24±0.0086
ML-KRR/ ML-KELM [24]	**0.0094±0.0001**	0.482±0.0062	0.924±0.0176	0.328±0.0056	0.036±0.0012
ML-BLS	0.0102±0.0001	0.3226±0.009	0.8744±0.0157	0.5145±0.0123	0.177±0.0036
ML-FBLS	0.01±0.0005	**0.1392±0.0056**	0.7526±0.0307	0.5844±0.0236	0.25±0.0018

As shown in Table 3.2 for the corel5k dataset, the optimum hamming loss is provided by ML-KRR, the optimum ranking loss is provided by ML-FBLS, optimum one error is provided by ML-ELM, optimum coverage is provided by BP-MLL, and optimum average precision is provided by ML-ELM.

Table 3.3: Comparison results performed on emotions dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.3297±0.017	0.4151±0.0244	0.5548±0.0598	3.1279±0.2145	**0.572±0.0245**
ML-ELM[57]	0.3607±0.038	0.9736±0.0042	0.5124±0.0208	0.4667±0.1826	0.0992±0.0454
BP-MLL[3]	0.3213±0.0144	0.4723±0.0247	0.6667±0.2887	0.5494±0.0236	0.5583±0.0187
ML-KRR/ ML-KELM [24]	0.6793±0.0095	0.3483±0.0165	0.5333±0.2173	0.4213±0.0246	0.4735±0.0268
ML-BLS	0.2234±0.0219	**0.1822±0.0319**	0.2432±0.0223	0.8187±0.061	0.4756±0.0472
ML-FBLS	**0.2031±0.0144**	0.2938±0.0186	**0.2232±0.0235**	0.9452±0.0213	0.5211±0.0321

As shown in Table 3.3 for the emotions dataset, ML-FBLS provides the optimum hamming loss, one error. ML-BLS provides the optimum ranking loss, ML-KRR provides optimum coverage, and RankSVM provides optimum average precision.

Table 3.4: Comparison results performed on enron dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0655±0.002	0.1189±0.002	0.4636±0.0246	0.2915±0.0091	0.5131±0.0105
ML-ELM[57]	0.0519±0.0011	0.6655±0.0277	0.3071±0.2959	0.6949±0.0689	0.6107±0.0177
BP-MLL[3]	0.0525±0.0016	0.2675±0.0301	0.6906±0.0286	**0.207±0.0089**	**0.7013±0.0092**
ML-KRR/ ML-KELM[24]	0.0619±0.0021	0.3572±0.026	0.6226±0.04	0.3955±0.0051	0.208±0.011
ML-BLS	0.0796±0.0019	0.2844±0.0189	0.6613±0.0641	0.6498±0.0217	0.1624±0.0214
ML-FBLS	**0.0502±0.0016**	**0.0997±0.007**	**0.2532±0.0275**	0.731±0.0344	0.1745±0.0216

As shown in Table 3.4 for the enron dataset, ML-FBLS provides optimum hamming loss, ranking loss, one error. BP-ML provides optimum coverage and average precision.

Table 3.5: Comparison results performed on medical dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0378±0.0011	0.1418±0.0083	0.728±0.0304	0.1619±0.0125	0.3987±0.0256
ML-ELM[57]	**0.0135±0.0011**	0.1667±0.0424	0.1082±0.0445	0.412±0.025	0.7299±0.0328
BP-MLL [3]	0.0192±0.0011	0.1213±0.037	0.7067±0.0575	**0.0456±0.016**	**0.7621±0.0198**
ML-KRR/ ML-KELM[24]	0.208±0.4047	0.2549±0.0179	0.6667±0.0272	0.0576±0.0173	0.3638±0.0507
ML-BLS	0.0161±0.0007	0.091±0.0336	0.3282±0.0967	0.1199±0.0439	0.4082±0.0731
ML-FBLS	0.0136±0.0017	**0.0321±0.0114**	**0.0209±0.0307**	0.3229±0.0923	0.4233±0.0575

As shown in Table 3.5 for the medical dataset, ML-ELM provides optimum hamming loss. ML-BLS provides optimum ranking loss coverage. ML-FBLS provides optimum one error. BP-MLL provides optimum average precision.

Table 3.6: Comparison results performed on scene dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.2718±0.0082	0.4844±0.0282	0.7786±0.0234	0.4807±0.096	0.4396±0.0173
ML-ELM[57]	0.1545±0.0741	0.4509±0.0948	0.2667±0.117	0.7605±0.3028	0.1645±0.2676
BP-MLL [3]	0.2839±0.0067	0.3159±0.0411	0.6±0.1491	0.3628±0.0124	0.4526±0.0074
ML-KRR/ ML-KELM[24]	0.767±0.0465	0.1675±0.0138	**0.0667±0.0913**	0.0889±0.0094	0.6733±0.0278
ML-BLS	0.0918±0.006	0.0818±0.0062	0.2134±0.0147	**0.0674±0.0313**	0.8136±0.0134
ML-FBLS	**0.0813±0.0039**	**0.0754±0.0077**	0.2064±0.0147	0.7433±0.0923	**0.8374±0.0155**

As shown in Table 3.6 for the scene dataset, ML-FBLS provides optimum hamming loss, ranking loss, and average precision. BP-MLL provides optimum one error. ML-BLS provides optimum coverage.

Table 3.7: Comparison results performed on yeast dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.2318±0.004	0.2113±0.0146	0.2486±6.7812	**0.2097±0.19**	0.703±0.0165
ML-ELM[57]	0.2121±0.0034	0.964±0.01	0.3277±0.0083	0.3429±0.1849	0.6676±0.0134
BP-MLL [3]	0.2078±0.0117	0.3315±0.0153	0.3571±0.1129	0.4596±0.0105	**0.7523±0.02**
ML-KRR/ ML-KELM[24]	**0.1941±0.0092**	0.2941±0.025	**0.1857±0.0639**	0.4328±0.006	0.5208±0.0214
ML-BLS	0.201±0.0073	0.3093±0.0083	0.2243±0.1129	0.9454±0.0204	0.4804±0.0132
ML-FBLS	0.1987±0.0089	**0.167±0.0082**	0.2201±0.0195	0.9532±0.0181	0.4941±0.0106

As shown in Table 3.7 for the yeast dataset, ML-KRR provides optimum hamming loss, one error. ML-FBLS provides optimum ranking loss. RankSVM provides optimum coverage. BP-MLL provides optimum average precision.

Table 3.8: Comparison results performed on bibtex dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0196±0.0002	0.329±0.0061	0.859±0.0035	0.774±0.0203	0.139±0.0013
ML-ELM[57]	**0.0148±0.0003**	0.6687±0.0194	**0.1847±0.0041**	0.6289±0.0213	0.3685±0.015
BP-MLL [3]	0.0162±0.0003	**0.0696±0.0039**	0.5384±0.0313	**0.0985±0.0038**	**0.548±0.0091**
ML-KRR/ ML-KELM [24]	0.0151±0.0001	0.4187±0.0099	0.6038±0.0118	0.3516±0.0075	0.1145±0.0064
ML-BLS	0.0231±0.0005	0.1902±0.004	0.3836±0.0544	0.5205±0.0188	0.4023±0.0079
ML-FBLS	0.021±0.0109	0.1755±0.0112	0.6878±0.0246	0.8532±0.0241	0.417±0.0074

As shown in Table 3.8, for the bibtex dataset, ML-ELM provides optimum hamming loss. BP-MLL provides optimum ranking loss, coverage, and average precision. ML-ELM provides optimum one error.

Table 3.9: Comparison results performed on rcv1v2s1 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0383±0.0024	0.1669±0.003	0.7697±0.0125	0.3137±0.0036	0.2509±0.0043
ML-ELM[57]	0.03±0.0007	0.1815±0.0055	0.5834±0.0536	0.6364±0.0207	0.3566±0.0222
BP-MLL[3]	0.0285±0.0003	0.6204±0.0317	0.7347±0.0308	0.565±0.0485	0.12±0.0159
ML-KRR/ ML-KELM [24]	0.0278±0.0004	0.3246±0.0109	0.8257±0.034	**0.2165±0.0046**	0.3889±0.0055
ML-BLS	**0.0281±0.0004**	0.0875±0.0081	0.4593±0.0777	0.4809±0.0323	0.4024±0.0125
ML-FBLS	0.0276±0.0006	**0.0445±0.0027**	**0.4235±0.0071**	0.544±0.0296	**0.4925±0.0104**

As shown in Table 3.9, for the rcv1v2s1, which is the first subset of the rcv1v2 dataset, ML-FBLS provides optimum hamming loss, ranking loss, one error, and average precision. ML-KELM provides optimum coverage.

Table 3.10: Comparison results performed on rcv1v2s2 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0365±0.0002	0.1652±0.003	0.7715±0.0109	0.3037±0.0587	0.2783±0.0156
ML-ELM[57]	0.0264±0.0004	0.1795±0.0165	0.5781±0.0377	0.6431±0.0217	0.4058±0.0069
BP-MLL[3]	0.0261±0.0002	0.6323±0.0278	0.7188±0.0772	**0.0772±0.0273**	0.0273±0.0146
ML-KRR/ ML-KELM[24]	0.0255±0.0003	0.4083±0.0102	0.802±0.049	0.1955±0.0097	0.4372±0.007
ML-BLS	0.0262±0.0006	0.097±0.0122	0.5354±0.0437	0.5299±0.0374	0.3714±0.0201
ML-FBLS	**0.0254±0.0009**	**0.0754±0.0025**	**0.4567±0.0129**	0.5735±0.0344	**0.4524±0.0109**

As shown in Table 3.10, for the rcv1v2s2, which is the second subset of the rcv1v2 dataset, ML-FBLS provides optimum hamming loss, ranking loss, one error, and average precision. ML-KELM provides optimum coverage. BP-MLL provides optimum coverage.

Table 3.11: Comparison results performed on rcv1v2s3 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0355±0.0025	0.1645±0.0049	0.7622±0.0103	0.304±0.0088	0.3077±0.0069
ML-ELM[57]	0.0259±0.0002	0.1753±0.0086	0.5809±0.0474	0.6224±0.0194	0.4147±0.0082
BP-MLL[3]	0.0259±0.0004	0.6246±0.0075	0.7267±0.0228	0.5392±0.0242	0.0893±0.0095
ML-KRR/ ML-KELM[24]	0.0254±0.0004	0.4251±0.0066	0.8178±0.0228	**0.1983±0.0078**	0.4398±0.0137
ML-BLS	0.0258±0.0007	0.0661±0.0099	0.5418±0.0568	0.3635±0.0334	0.3898±0.0181
ML-FBLS	**0.0242±0.0007**	**0.0485±0.0019**	**0.4245±0.0121**	0.6189±0.0205	**0.4452±0.0154**

As shown in Table 3.11, for the rcv1v2s3, which is the third subset of the rcv1v2 dataset, ML-FBLS provides optimum hamming loss, ranking loss, one error, and average precision. ML-KELM provides optimum coverage. ML-KRR provides optimum coverage.

Table 3.12: Comparison results performed on rcv1v2s4 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0295±0.0005	0.141±0.0033	0.746±0.0144	0.2689±0.0044	0.3516±0.009
ML-ELM[57]	0.0236±0.0004	0.1827±0.0123	0.5679±0.0611	0.5941±0.023	0.4752±0.0137
BP-MLL[3]	0.0246±0.0003	0.5784±0.0254	0.804±0.053	0.5167±0.0588	0.0865±0.0187
ML-KRR/ ML-KELM[24]	0.0238±0.0003	0.4251±0.011	0.7663±0.0435	**0.1701±0.0048**	**0.4996±0.009**
ML-BLS	0.0295±0.0317	0.1657±0.11	0.5215±0.4528	0.3703±0.6085	0.4136±0.1575
ML-FBLS	**0.0217±0.0013**	**0.11±0.0045**	**0.4528±0.0121**	0.6085±0.028	0.4575±0.0096

As shown in Table 3.12, for the rcv1v2s4, which is the fourth subset of the rcv1v2 dataset, ML-FBLS provides optimum hamming loss, ranking loss, and one error. ML-KRR provides optimum coverage and average precision.

Table 3.13: Comparison results performed on rcv1v2s5 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0361±0.0004	0.1574±0.0024	0.7528±0.0109	0.2993±0.0041	0.3113±0.0167
ML-ELM[57]	0.0259±0.0005	0.1853±0.0085	0.59±0.0526	0.6331±0.0237	0.4416±0.01
BP-MLL[3]	0.0262±0.0002	0.5892±0.0232	0.7941±0.0608	0.511±0.0202	0.1045±0.0125
ML-KRR/ ML-KELM[24]	0.0258±0.0003	0.4169±0.0059	0.8376±0.0325	**0.185±0.0039**	0.4567±0.0115
ML-BLS	0.0241±0.0008	0.0606±0.0053	0.4691±0.0452	0.3538±0.0175	0.4094±0.0084
ML-FBLS	**0.0237±0.0006**	**0.0462±0.0023**	**0.4227±0.0116**	0.4434±0.026	**0.4643±0.024**

As shown in Table 3.13, for the rcv1v2s5, which is the fifth subset of the rcv1v2 dataset, ML-FBLS provides optimum hamming loss, ranking loss, one error, and average precision. ML-KELM provides optimum coverage. ML-KRR provides optimum coverage and average precision.

Table 3.14: Average of performance measures on multi-label datasets

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.09525±0.00321	0.2202±0.0086	0.6842±0.5833	0.571525±0.0523	0.3728±0.0130
ML-ELM[57]	0.0791±0.010	0.4462±0.0221	0.4153±0.0625	0.6091±0.0756	0.4156±0.0388
BP-MLL[3]	0.0868±0.0031	0.408±0.0232	0.6890±0.0771	0.3495±0.021	0.3701±0.0136
ML-KRR/ ML-KELM[24]	0.1719±0.0394	0.3602±0.0133	0.6376±0.0542	**0.2534±0.0088**	0.3843±0.016
ML-BLS	0.0649±0.0060	0.1615±0.02145	0.4546±0.088	0.4778±0.0782	0.4088±0.0330
ML-FBLS	**0.0583±0.0038**	**0.1089±0.0069**	**0.3787±0.01891**	0.6601±0.0370	**0.4524±0.0180**

It is difficult for a single algorithm to perform better than all other algorithms for five evaluation measures, so the average of the five evaluation measure on twelve datasets is shown in Table 3.14. ML-FBLS provides optimum hamming loss, ranking

loss, one error, and average precision on an average of five evaluation measures. ML-KRR provides optimum coverage. Further, we perform the Wilcoxon signed-rank test to test the significance of performance of proposed approaches ML-BLS and ML-FBLS, which are discussed in the next section.

3.6 Statistical Analysis

In this section, we discuss the statistical analysis of the results using the Wilcoxon signed-rank test provided by the experimentation results. Wilcoxon signed-rank test is used to verify the significance of performance between each pair of compared multi-label approaches. Wilcoxon signed-rank test considers the difference between the evaluation measures provided by the pair of classification approaches[133].

Table 3.15: Statistics of p-values obtained from Wilcoxon's signed rank test for hamming loss

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-BLS	ML-FBLS
ML-BLS	0.00222	0.03318	0.5552	0.93624	–	0.00222
ML-FBLS	0.00374	0.0232	0.1878	0.9894	0.00222	–

Table 3.16: Statistics of p-values obtained from Wilcoxon's signed rank test for ranking loss

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-BLS	ML-FBLS
ML-BLS	0.30772	0.00222	0.0232	0.00288	–	0.01878
ML-FBLS	0.00222	0.00222	0.00374	0.00222	0.01878	–

Table 3.17: Statistics of p-values obtained from Wilcoxon's signed rank test for one error

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-BLS	ML-FBLS
ML-BLS	0.00758	0.93624	0.00222	0.01208	–	0.0232
ML-FBLS	0.00222	0.20766	0.00374	0.0096	0.0232	–

Table 3.18: Statistics of p-values obtained from Wilcoxon's signed rank test for coverage

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-BLS	ML-FBLS
ML-BLS	0.00222	0.13622	0.18352	0.00288	–	0.00222
ML-FBLS	0.034	0.93624	0.0048	0.0222	0.00222	–

Table 3.19: Statistics of p-values obtained from Wilcoxon's signed rank test for average precision

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-BLS	ML-FBLS
ML-BLS	0.38978	0.477	0.5823	0.93624	–	0.00222
ML-FBLS	0.18352	0.5287	0.30772	0.0601	0.00222	–

Tables 3.15, 3.16, 3.17, 3.18, and 3.19 denote the p-value obtained by conducting Wilcoxon signed-rank test on the basis of significance level 0.05 for five evaluation measures. The Wilcoxon signed-rank test is performed between each pair of compared approaches RankSVM, ML-ELM, BP-MLL, ML-KRR/ML-KELM, ML-BLS, and ML-FBLS based on the evaluation measures hamming loss, ranking loss, one error, coverage, and average precision. The terms $--$ in tables 3.15, 3.16, 3.17, 3.18, and 3.19 denote that there is no self comparison is possible based on the Wilcoxon signed-rank test. The ML-FBLS provides better performance in comparison to ML-BLS in the majority of the evaluation measures. The Wilcoxon signed-rank test performed based on evaluation measures indicates that BLS based multi-label approaches provide significant results. BLS based multi-label classifiers are better than other approaches, and ML-FBLS provide better significant results in comparison to other compared approaches.

3.7 Analysis of Computational Complexity

There are two main steps in the ML-FBLS procedure during training. The first step is to compute the parameters and hyperparameters of the neuro-fuzzy step, and the second is to compute the parameters of the threshold function. In the first step there are the several fuzzy subsystems which are generated with

$O\left(N\sum_{i=1}^{n}K_i\sum_{j=1}^{m}L_j\right)$ and the fuzzy rules in each fuzzy subsystem are computed with $O\left(NM\sum_{i=1}^{n}K_i^2\right)$ and the enhancement nodes computation complexity is computed as $O\left(\sum_{i=1}^{n}K_i+\sum_{j=1}^{m}L_j\right)^3$. The threshold step can be computed with $O\left(N^2Q\right)$. The complete complexity can be shown as follows.

$$O\left(N\sum_{i=1}^{n}K_i\sum_{j=1}^{m}L_j+NM\sum_{i=1}^{n}K_i^2+\left(\sum_{i=1}^{n}K_i+\sum_{j=1}^{m}L_j\right)^3+N^2Q\right). \tag{3.25}$$

3.8 Summary

In this chapter, we have proposed two multi-label classification approaches referred to as ML-BLS and ML-FBLS. The ML-BLS is designed by enhancing the traditional BLS algorithm by adding an adaptive threshold function at the output layer. This adapting threshold function considers the data instances for finding the threshold to separate the relevant and irrelevant labels for a particular instance. To provide better interpretability to ML-BLS, we have also proposed the hybrid neuro-fuzzy enhancement named as ML-FBLS by incorporating fuzzy rules in ML-BLS. The ML-BLS and ML-FBLS are non-iterative approaches that use pseudoinverse to compute the parameters. The performance of ML-BLS and ML-FBLS is verified with the experimental evaluations performed on twelve benchmark datasets of various domains. The proposed algorithms lead to better results in comparison to state-of-the-art multi-label algorithms in terms of hamming loss, ranking loss, and one error evaluation metrics. The proposed approaches, ML-BLS and ML-FBLS, have the limitation that these approaches are unable to provide optimum coverage and average precision evaluation measure for the majority of the datasets. This limitation is tackled in the next chapter by proposing simple and effective Random Vector Functional Link Network based multi-label classifiers.

Chapter 4

Random Vector Functional Link Neural Network based Multi-Label Classification

In this chapter, we have proposed two algorithm adaptation multi-label classifiers based on Random Vector Functional Link Neural Network (RVFL) named ML-RVFL and ML-KRVFL. These multi-label neural networks, ML-RVFL and ML-KRVFL are Single Layer Feedforward Neural Networks (SLFN). The ML-RVFL is a simple non-iterative neural network in comparison to ML-BLS and ML-FBLS. The ML-RVFL classifier improves the overall performance of ML-BLS and ML-FBLS in terms of five evaluation measures. The ML-RVFL needs information of the number of neurons in the enhancement layer. This information is not needed if we contain the kernelized RVFL for multi-label classification. To avoid the estimation of neurons in the hidden layer, we have proposed the ML-KRVFL, which is based on the kernel function. The ML-RVFL and ML-KRVFL use the adaptive threshold method to compute the threshold during training which makes these approaches to consider the multi-label data. The experimental results are analyzed using the Friedman ranking and Wilcoxon signed-rank tests. The proposed multi-label classifiers ML-RVFL and ML-KRVFL outperform other state-of-the-art multi-label classifiers in terms of five evaluation metrics. The introduction of multi-label classification using non-iterative approaches is discussed in the next section.

4.1 Introduction

The SLFN is designed using an input layer, a hidden layer, and an output layer. The parameters of SLFN are initialized randomly and updated using the back-propagation algorithm. The convergence of back-propagation based SLFN is time-consuming, so the non-iterative procedure based SLFN has been proposed to provide a fast training procedure with efficient performance[26]. In recent years various non-iterative approaches have been proposed for multi-label classification, such as ML-ELM [57] and ML-KELM/ML-KRR[24]. These classifiers are fast and efficient neural networks for multi-label classification. These approaches lack direct connections between the input layer and output layer. The direct connections are useful to provide information of the input features in the neural networks. The multi-label classifiers ML-BLS and ML-FBLS contain the direct links between the input and output layer. The ML-BLS converts the inputs to the mapped features and then creates the enhancement nodes with the help of mapped features. The ML-FBLS replaces the mapped features with fuzzy systems to get better interpretability of the results. The RVFL [27] is another non-iterative single-layer neural network that connects the inputs to the output layer using direct links and is similar to ELM. The RVFL is illustrated in Figure 2.1. The RVFL is able to solve the traditional multi-class classification. We have proposed the adaptation of RVFL and kernelized RVFL for multi-label classification, which provides the optimum evaluation metrics. The details and formulation of the proposed ML-RVFL and ML-KRVFL are discussed in the next sections.

4.2 Multi-Label Classifier based on Random Vector Functional Link Neural Network: ML-RVFL

In this section, we have discussed the proposed ML-RVFL and its mathematical formulation. The proposed ML-RVFL is a single-layer feedforward neural network

that uses pseudoinverse to compute the parameters of the classifier. In ML-RVFL, the weights and bias of the hidden layer are initialized randomly within a suitable range. Only the output weights of the output layer are computed using pseudoinverse. The

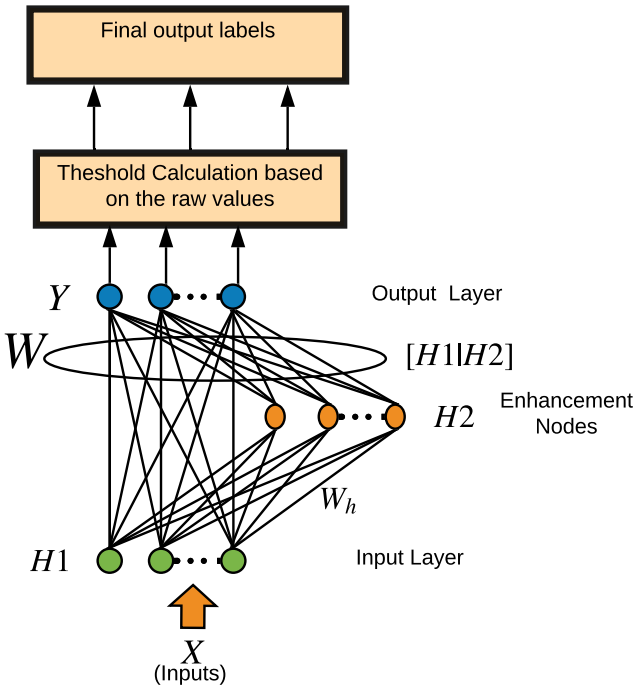


Figure 4.1: Multi-label Random Vector Functional Link Neural Network

RVFL has universal approximation capability because of its neural network based architecture. Hence it can be directly applied to multiple problems. As shown in Figure 4.1 the input layer $H1$ consists the original input features $X \in R^{N \times M}$. The input layer is transformed into a nonlinear transformation using enhancement layer $H2$. Input $H1$ and enhancement layer $H2$ can be concatenated as follows

$$H = [H1|H2] \tag{4.1}$$

Further the RVFL can be denoted by the following mathematical expression

$$HW = Y \tag{4.2}$$

The size of matrix H1 is $N \times M$, and the size of H2 is $N \times L$, where the N is the number of instances in training data, M denotes the size of the feature vector of the input instance, and L is the number of neurons in hidden or enhancement layer. Hence The size of matrix $H = [H1|H2]$ is $N \times (M+L)$. The output weights can be computed using the analytical or normal solution as follows

$$W = H^{+}Y \tag{4.3}$$

The RVFL can be represented as an optimization problem using following mathematical expression

$$\arg\min_{\mathbf{W}} : \|\boldsymbol{HW} - \boldsymbol{Y}\|_v^{\sigma_1} + \lambda\|\boldsymbol{W}\|_u^{\sigma_2} \tag{4.4}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, $u > 0$, and $v > 0$ is a kind of norm regularization and λ is regularization constant. Above expression becomes a l_2 convex optimization by taking $\sigma_1 = \sigma_2 = u = v = 2$. The primal space solution for W is defined as below

$$W = (H^T H + \lambda I)^{-1} H^T Y \tag{4.5}$$

We can transform the above primal space solution into the dual space solution as follows:

$$H^T(HH^T + \lambda I)^{-1} = (H^T H + \lambda I)^{-1} H^T \tag{4.6}$$

$$\begin{aligned} &(H^T H + \lambda I)^{-1}(H^T H + \lambda I)H^T(HH^T + \lambda I)^{-1}Y \\ &= (H^T H + \lambda I)^{-1} H^T (HH^T + \lambda I)(HH^T + \lambda I)^{-1} Y \end{aligned} \tag{4.7}$$

$$H^T(HH^T + \lambda I)^{-1}Y = (H^T H + \lambda I)^{-1} H^T Y \tag{4.8}$$

$$H^T(HH^T + \lambda I)^{-1}Y = W \tag{4.9}$$

so dual space solution is defined as the following mathematical expression

$$W = H^T(HH^T + \lambda I)^{-1}Y \tag{4.10}$$

Depending on the number of training samples or total feature dimensions (i.e., input features plus the total number of hidden neurons), dual or primal solution can be used to reduce the complexity of the matrix inversion [28]. The primal and dual space solution [28] of the above problem can be used according to the following equations:

$$W = \begin{cases} (H^T H + \lambda I)^{-1} H^T Y, & \text{primal space} \\ H^T (H H^T + \lambda I)^{-1} Y, & \text{dual space} \end{cases} \tag{4.11}$$

where primal space solution refers to $N < (M + L)$ and dual space solution refers to $N > (M + L)$. Using the RVFL network from Eq.(4.3) and Eq.(4.11), the raw output at the layer Y of RVFL can be written as:

$$f(x) = HW \tag{4.12}$$

The solution of primal and dual space can be used to compute the $f(x)$ as mentioned in [28]. By substituting the value of W from Eq.(4.11) to Eq.(4.12), $f(x)$ can be written in dual space as

$$f(x) = H \left(H^T H + \lambda I \right)^{-1} H^T Y. \tag{4.13}$$

The result computed by the Eq.(4.13) is provided to compute the threshold computation to create ML-RVFL for multi-label classification, which is discussed in section 3.4. In the next section, we have proposed the kernelized version of ML-RVFL.

4.3 Multi-Label Kernelized Random Vector Functional Link Network: ML-KRVFL

The structure of RVFL, its fast learning, and its generalization capability are useful in various applications. Kernel based RVFL is developed to avoid the selection of hidden mapping and the number of hidden neurons in RVFL. By substituting the value

of H from Eq.(4.1) in Eq.(4.13), the output $f(x)$ of ML-KRVFL can be represented as follows:

$$f(x) = [h_1|h_2]\left[\frac{H1}{H2}\right]^T \left([H1|H2]\left[\frac{H1}{H2}\right]^T + \lambda I\right)^{-1} Y \tag{4.14}$$

where h_1 denotes the i^{th} instance of input matrix X and h_2 denoted mapped feature map of the i^{th} instance of input matrix X. The Eq. (4.14) can be rewritten as follows:

$$f(x) = \left(\left[h_1 H1^T\right] + \left[h_2 H2^T\right]\right)\left(\left[H1H1^T\right] + \left[H2H2^T\right] + \lambda I\right)^{-1} Y \tag{4.15}$$

The kernel matrix for the Eq.(4.15) can be defined as follows:

$$\Omega = H1H1^T and \left[h_1 H1^T\right] = K(x_i, x_l) \tag{4.16}$$

$$\tilde{\Omega} = H2H2^T and \left[h_2 H2^T\right] = \tilde{K}(x_i, x_l) \tag{4.17}$$

where $l \in 1, ..., L$. The output of KRVFL can be computed as following mathematical expression:

$$f(x) = \begin{bmatrix} k(x_i, x_1) \\ . \\ . \\ . \\ k(x_i, x_L) \end{bmatrix} + \begin{bmatrix} \tilde{k}(x_i, x_1) \\ . \\ . \\ . \\ \tilde{k}(x_i, x_L) \end{bmatrix} \left(\Omega + \tilde{\Omega} + \lambda I\right)^{-1} Y \tag{4.18}$$

K is a linear kernel and $\tilde{K}$ is the radial basis function $exp^{\frac{-||x-x_i||^2}{2\sigma^2}}$ used in experimentation. The σ is known as kernel parameter . The result computed by the Eq.(4.18) is provided to compute the threshold computation to create ML-KRVFL for multi-label classification which is discussed in section 3.4. The ML-RVFL and ML-KRVFL are described in Algorithm 3 and 4 respectively.

Algorithm 3: Multi-label Random Vector Functional Link Network

Input : Training samples $(X, Y) \in R^{(N \times (M+Q))}$
Output: A ML-RVFL with output weight W, threshold parameters w and b

1 Initialize hidden layer neuron number L, activation function , regularization parameter λ
2 Pre-process: target values of training set are converted to bipolar representation (-1 and 1);
3 Assign input weight W_h and bias for hidden layer $H2$ randomly
4 Compute matrix H using concatenation of $H1$ and $H2$
5 Compute output weights W using Eq.(4.11);
6 Using weight matrix W, Compute raw values $f(x)$ at output layer Y for input sample x
7 Compute threshold function parameters w and b using Eq.(3.22) and Eq.(3.23) as follows. $min_{w,b} \sum_{i=1}^{n}((w^T.c(x_i)) + b - t_{x_i})^2$.
8 Calculate outputs $c(x)$ and threshold for each test data instance according to Eq.(3.22) and Eq.(3.23)
9 Compare $c(x)$ with a threshold value and obtain multi-label identification final outputs according to Eq.(3.24).

Algorithm 4: Multi-label Kernel Random Vector Functional Link Network

Input : Training samples $(X, Y) \in R^{(N \times (M+Q))}$
Output: A ML-KRVFL with output weight W, threshold parameters w and b

1 Initialize kernel parameters σ and regularization cost parameter λ
2 Pre-process: Conversion of output label values of training set to bipolar values 1 or -1;
3 Compute kernel matrix Ω and $\tilde{\Omega}$
4 Using weight matrix W, compute raw values $f(x)$ at output layer Y for input sample x using Eq.(4.18)
5 Compute threshold function parameters w and b using Eq.(3.22) and Eq.(3.23) as $min_{w,b} \sum_{i=1}^{n}((w^T.c(x_i)) + b - t_{x_i})^2$.
6 Calculate outputs $c(x)$ and threshold for each test data instance according to Eq.(3.22) and Eq.(3.23)
7 Compare $c(x)$ with a threshold value and obtain multi-label identification final outputs according to Eq.(3.24)

4.4 Experimentation and Results

In this section, the results of experiments are discussed to verify the effectiveness of the proposed multi-label non-iterative approaches ML-RVFL and ML-KRVFL. Multi-label classification experimentation is carried out on 12 benchmarks multi-label

Table 4.1: Optimal parameter settings for multi-label classifiers

	BP-MLL[3]	ML-ELM[57]		ML-KRR/ ML-KELM [24]		ML-RVFL		ML-KRVFL		ML-BLS			ML-FBLS		
	Hidden layer nodes	Hidden layer nodes	C=$\frac{1}{\lambda}$	σ	C=$\frac{1}{\lambda}$	optimal hidden nodes	λ	σ	λ	N_f	N_m	N_e	N_r	N_t	N_e
Bibtex	367	120	0.5	2^{-2}	2^{1}	100	0.7	2^{-2}	1	51	47	44	10	3	2
Corel5k	100	250	0.5	2^{-2}	2^{1}	250	0.5	2^{-1}	0.8	10	14	55	1	12	19
Emotions	15	60	0.5	2^{-2}	2^{0}	60	0.5	2^{-1}	0.8	51	10	13	88	5	102
Enron	200	120	0.5	2^{-2}	2^{0}	150	0.7	2^{-2}	0.8	36	26	28	67	3	19
Scene	59	85	0.5	2^{-2}	2^{0}	85	0.7	2^{-1}	1	25	18	37	23	97	79
Yeast	21	100	0.5	2^{-2}	2^{0}	80	0.5	2^{-2}	0.8	10	10	35	89	39	93
Medical	290	120	0.5	2^{-2}	2^{1}	200	0.5	2^{-2}	0.8	24	7	58	23	97	79
rcv1v2-s1	9448	120	1	2^{-2}	2^{1}	100	0.9	2^{-2}	1	64	66	63	44	33	35
rcv1v2-s2	9448	120	1	2^{-2}	2^{1}	100	0.9	2^{-2}	1	39	55	35	39	7	35
rcv1v2-s3	9448	120	1	2^{-2}	2^{1}	100	0.9	2^{-2}	1	59	6	16	41	26	35
rcv1v2-s4	9448	120	1	2^{-2}	2^{1}	100	0.9	2^{-2}	1	29	35	53	50	9	21
rcv1v2-s5	9448	120	1	2^{-2}	2^{1}	100	0.9	2^{-2}	1	34	13	33	40	35	33

datasets[1] [2], which are described in Table 2.6. Comparative results based on evaluation metrics are shown in Table 4.2 to Table 4.13. All the experiments related to the comparison of multi-label classification are performed on MATLAB 2017a. Five-fold cross-validation is carried out on all 12 multi-label datasets. For each dataset, we have used five-fold cross-validation. In each fold, 1 set is used for testing, and the rest four sets are used as the training set. For this purpose, we have generated the indexing of instances for each multi-label dataset for each fold. The indexing for each fold is the same for all the compared approaches. Using this five-fold cross-validation, 80% of the total data is used as training data and 20% for tests for all the datasets. We compute the hamming loss, one error, ranking loss, coverage, and average precision to compute the effectiveness of proposed ML-RVFL and ML-KRVFL approaches. We use the same experimental setup, parameter settings, and state-of-the-art approaches for comparison purposes which are discussed in Chapter 3.

4.4.1 Experimental Setup

The parameters range and optimal parameters of the compared approaches are the same as discussed in Chapter 3. We also compare the performance of ML-RVFL and ML-KRVFL with the approaches ML-BLS and ML-FBLS, which are proposed in chapter 3. For the ML-RVFL, the regularization parameter λ is searched between 0.1 and 1 with the interval of 0.2. The activation function used in ML-RVFL and

[1]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

[2]http://mulan.sourceforge.net/datasets-mlc.html

ML-KELM is sigmoid, and for BP-MLL, the activation function used is tanh, as it is mentioned in[3]. For the ML-KRVFL approach, kernel parameter σ is taken as 2^x where $x = [-8, -6, \ldots, 0]$. The regularization parameter λ for the ML-KRVFL is searched between 0.1 to 1. The regularization parameter λ for the ML-RVFL is searched by fixing the hidden nodes of ML-RVFL by keeping near the values provided by the ML-ELM for the corresponding datasets. For the ML-KRVFL, the regularization parameter λ is searched by fixing the kernel parameter σ of ML-KRVFL. The values of σ for ML-KRVFL are also considered near the values of σ for ML-KRR/ML-KELM to compare the performance of the classifiers. Table 4.1 describes the optimal parameter settings used in the experimentation for the compared approaches and proposed approaches.

4.4.2 Results and Performance Evaluation

In this section, we provide the experimental results performed on the twelve multi-label datasets from various domains as mentioned in 2.6. The optimum performance is highlighted in bold for each evaluation metric in table 4.2 to 4.13. Further, the average performance on twelve datasets for five evaluation metrics is described in table 4.14. We also include the ML-BLS and ML-FBLS approaches for comparison, which are introduced in chapter 3.

Table 4.2: Comparison results performed on corel5k dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0109±0.0001	0.147±0.0034	0.776±0.0184	0.323±0.0056	0.209±0.0085
ML-ELM[57]	0.0104±0.0001	0.561±0.0154	0.377±0.0081	0.875±0.0138	0.254±0.0134
BP-MLL[3]	0.0095±0	0.273±0.0109	0.931±0.0129	0.263±0.0053	0.24±0.0086
ML-KRR/ ML-KELM [24]	**0.0094±0.0001**	0.482±0.0062	0.924±0.0176	0.328±0.0056	0.036±0.0012
ML-BLS	0.0102±0.0001	0.3226±0.009	0.8744±0.0157	0.5145±0.0123	0.177±0.0036
ML-FBLS	0.01±0.0005	0.1392±0.0056	0.7526±0.0307	0.5844±0.0236	0.25±0.0018
ML-RVFL	0.01±0	0.112±0.0029	0.484±0.0115	0.281±0.0201	0.2838±0.0067
ML-KRVFL	0.0112±0.0001	**0.0909±0.009**	**0.3494±0.0112**	**0.2445±0.0099**	**0.3172±0.0039**

Corel5k is an image domain dataset that contains samples from a museum, web archive images, and newspaper images. As mentioned in Table 4.2, ML-KRR/ML-KELM provides optimum hamming loss for the corel5k dataset. ML-KRVFL provides

optimum ranking loss, one error, coverage, and average precision. The corel5k has 499 features, 374 labels, and 5000 instances.

Table 4.3: Comparison results performed on emotions dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.3297±0.017	0.4151±0.0244	0.5548±0.0598	3.1279±0.2145	0.572±0.0245
ML-ELM[57]	0.3607±0.038	0.9736±0.0042	0.5124±0.0208	0.4667±0.1826	0.0992±0.0454
BP-MLL[3]	0.3213±0.0144	0.4723±0.0247	0.6667±0.2887	0.5494±0.0236	0.5583±0.0187
ML-KRR/ ML-KELM [24]	0.6793±0.0095	0.3483±0.0165	0.5333±0.2173	0.4213±0.0246	0.4735±0.0268
ML-BLS	0.2234±0.0219	0.1822±0.0319	0.2432±0.0223	0.8187±0.061	0.4756±0.0472
ML-FBLS	0.2031±0.0144	0.2938±0.0186	0.2232±0.0235	0.9452±0.0213	0.5211±0.0321
ML-RVFL	0.2097±0.0256	0.168±0.0316	0.25±0.0319	0.3978±0.0916	0.7004±0.0504
ML-KRVFL	**0.1856±0.0192**	**0.136±0.0292**	**0.2115±0.0235**	**0.2928±0.0338**	**0.7947±0.0242**

The emotions dataset contains the music samples with 593 instances of 6 kinds. This is a small dataset. Table 4.3 describes the evaluation metrics performed on emotions dataset. The proposed approaches ML-RVFL, ML-KRVFL, ML-BLS, and ML-FBLS provide optimum hamming loss for the emotions dataset. ML-KRVFL provides the best hamming loss, ranking loss, one error, coverage, and average precision for the emotions datasets.

Table 4.4: Comparison results performed on enron dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0655±0.002	0.1189±0.002	0.4636±0.0246	0.2915±0.0091	0.5131±0.0105
ML-ELM[57]	0.0519±0.0011	0.6655±0.0277	0.3071±0.2959	0.6949±0.0689	0.6107±0.0177
BP-MLL[3]	0.0525±0.0016	0.2675±0.0301	0.6906±0.0286	0.207±0.0089	**0.7013±0.0092**
ML-KRR/ ML-KELM[24]	0.0619±0.0021	0.3572±0.026	0.6226±0.04	0.3955±0.0051	0.208±0.011
ML-BLS	0.0796±0.0019	0.2844±0.0189	0.6613±0.0641	0.6498±0.0217	0.1624±0.0214
ML-FBLS	0.0502±0.0016	0.0997±0.007	0.2532±0.0275	0.731±0.0344	0.1745±0.0216
ML-RVFL	0.0473±0.0021	0.0831±0.0047	0.2344±0.0257	0.2069±0.027	0.4276±0.015
ML-KRVFL	**0.022±0.0148**	**0.048±0.0105**	**0.2123±0.0208**	**0.2058±0.0236**	0.4606±0.017

As mentioned in Table 4.4, ML-KRVFL provides optimum hamming loss, ranking loss, one error, and coverage for enron dataset. BP-MLL provides optimum average precision for the enron dataset.

Table 4.5: Comparison results performed on medical dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0378±0.0011	0.1418±0.0083	0.728±0.0304	0.1619±0.0125	0.3987±0.0256
ML-ELM[57]	0.0135±0.0011	0.1667±0.0424	0.1082±0.0445	0.412±0.025	0.7299±0.0328
BP-MLL [3]	0.0192±0.0011	0.1213±0.037	0.7067±0.0575	0.0456±0.016	**0.7621±0.0198**
ML-KRR/ ML-KELM[24]	0.208±0.4047	0.2549±0.0179	0.6667±0.0272	0.0576±0.0173	0.3638±0.0507
ML-BLS	0.0161±0.0007	0.091±0.0336	0.3282±0.0967	0.1199±0.0439	0.4082±0.0731
ML-FBLS	0.0136±0.0017	0.0321±0.0114	0.0209±0.0307	0.3229±0.0923	0.4233±0.0575
ML-RVFL	0.014±0.001	0.0222±0.0079	0.0129±0.0133	0.0978±0.0445	0.6977±0.596
ML-KRVFL	**0.0116±0.043**	**0.021±0.0132**	**0.0118±0.0329**	**0.0901±0.0165**	0.6677±0.026

Medical datasets contain the text corpus related to a substantial proportion of pediatric radiology activity. Table 4.5 describes the evaluation metrics performed on the medical dataset. ML-KRVFL provides optimum hamming loss, ranking loss, one error, and coverage. BP-MLL provides optimum average precision for the medical dataset.

Table 4.6: Comparison results performed on scene dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.2718±0.0082	0.4844±0.0282	0.7786±0.0234	0.4807±0.096	0.4396±0.0173
ML-ELM[57]	0.1545±0.0741	0.4509±0.0948	0.2667±0.117	0.7605±0.3028	0.1645±0.2676
BP-MLL [3]	0.2839±0.0067	0.3159±0.0411	0.6±0.1491	0.3628±0.0124	0.4526±0.0074
ML-KRR/ ML-KELM[24]	0.767±0.0465	0.1675±0.0138	**0.0667±0.0913**	0.0889±0.0094	0.6733±0.0278
ML-BLS	0.0918±0.006	0.0818±0.0062	0.2134±0.0147	**0.0674±0.0313**	0.8136±0.0134
ML-FBLS	0.0813±0.0039	**0.0754±0.0077**	0.2064±0.0147	0.7433±0.0923	**0.8374±0.0155**
ML-RVFL	**0.0911±0.0061**	0.0764±0.0089	0.2198±0.0217	0.6658±0.0372	0.8133±0.0142
ML-KRVFL	0.119±0.005	0.1187±0.0107	0.2995±0.0194	0.1146±0.01	0.8126±0.0142

Scene datasets contain 2407 multi-label images with six possible labels. Table 4.6 describes the evaluation metrics performed on scene dataset. ML-RVFL provides optimum hamming loss, ML-BLS provides optimum coverage, and ML-KRR/ML-KELM provides optimum one error for scene dataset. ML-FBLS provides optimum ranking loss and average precision.

Table 4.7: Comparison results performed on yeast dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.2318±0.004	0.2113±0.0146	0.2486±6.7812	**0.2097±0.19**	0.703±0.0165
ML-ELM[57]	0.2121±0.0034	0.964±0.01	0.3277±0.0083	0.3429±0.1849	0.6676±0.0134
BP-MLL [3]	0.2078±0.0117	0.3315±0.0153	0.3571±0.1129	0.4596±0.0105	0.7523±0.02
ML-KRR/ ML-KELM[24]	0.1941±0.0092	0.2941±0.025	**0.1857±0.0639**	0.4328±0.006	0.5208±0.0214
ML-BLS	0.201±0.0073	0.3093±0.0083	0.2243±0.1129	0.9454±0.0204	0.4804±0.0132
ML-FBLS	0.1987±0.0089	0.167±0.0082	0.2201±0.0195	0.9532±0.0181	0.4941±0.0106
ML-RVFL	**0.1712±0.0069**	0.1545±0.0095	0.2121±0.152	0.6543±0.128	0.6801±0.0095
ML-KRVFL	0.1735±0.0091	**0.1468±0.0173**	0.2062±0.0267	0.5176±0.0064	**0.7542±0.0111**

Yeast dataset denotes the 14 possible functional classes as an output for each genome sequence represented by its instances. Table 4.7 describes the evaluation metrics performed on the yeast dataset. ML-RVFL provides optimum hamming loss for yeast dataset. ML-KRVFL provides optimum ranking loss and average precision. ML-KRR/ML-KELM provides optimum one error, and RankSVM provides optimum coverage for yeast dataset. For the yeast dataset, The ML-KRVFL is performing the best ranking loss, as the yeast dataset is the genome-related dataset and the ranking loss is in focus for optimizing for yeast-related work because of the objective of the genome data properties [3].

Table 4.8: Comparison results performed on bibtex dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0196±0.0002	0.329±0.0061	0.859±0.0035	0.774±0.0203	0.139±0.0013
ML-ELM[57]	0.0148±0.0003	0.6687±0.0194	**0.1847±0.0041**	0.6289±0.0213	0.3685±0.015
BP-MLL [3]	0.0162±0.0003	**0.0696±0.0039**	0.5384±0.0313	**0.0985±0.0038**	0.548±0.0091
ML-KRR/ ML-KELM [24]	0.0151±0.0001	0.4187±0.0099	0.6038±0.0118	0.3516±0.0075	0.1145±0.0064
ML-BLS	0.0231±0.0005	0.1902±0.004	0.3836±0.0544	0.5205±0.0188	0.4023±0.0079
ML-FBLS	0.021±0.0109	0.1755±0.0112	0.6878±0.0246	0.8532±0.0241	0.417±0.0074
ML-RVFL	**0.0132±0.0004**	0.077±0.0061	0.3671±0.0112	0.5482±0.0304	0.4043±0.0104
ML-KRVFL	0.0145±0.0005	0.106±0.0083	0.3601±0.0118	0.2023±0.012	**0.5744±0.0099**

Table 4.8 describes the evaluation metrics performed on bibtex dataset. ML-ELM provides optimum one error, BP-MLL provides optimum ranking loss and coverage. ML-RVFL and ML-KRVFL provide optimum hamming loss and average precision, respectively. Bibtex is a text dataset related to the recommender system, and the related evaluation metric for the cases are precision, recall, and F measure mentioned in[17]. We have used this dataset for comparison purposes and demonstrate the limi-

tation of the proposed approach. In text-related datasets, the embedding of the labels is an important consideration for semantic representation. This opens the scope of the future research of non-iterative learning for the semantic consideration where the labels can be represented as feature vectors of their properties.

Datasets rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5 are related to text domain. These datasets are manually categorized from newswire stories which are recently made available by Reuters, Ltd. Table 4.9 to Table 4.13 shows the evaluation results for rcv1v2s1, rcv1v2s2, rcv1v2s3, rcv1v2s4, and rcv1v2s5 datssets.

Table 4.9: Comparison results performed on rcv1v2s1 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0383±0.0024	0.1669±0.003	0.7697±0.0125	0.3137±0.0036	0.2509±0.0043
ML-ELM[57]	0.03±0.0007	0.1815±0.0055	0.5834±0.0536	0.6364±0.0207	0.3566±0.0222
BP-MLL[3]	0.0285±0.0003	0.6204±0.0317	0.7347±0.0308	0.565±0.0485	0.12±0.0159
ML-KRR/ ML-KELM [24]	0.0278±0.0004	0.3246±0.0109	0.8257±0.034	**0.2165±0.0046**	0.3889±0.0055
ML-BLS	0.0281±0.0004	0.0875±0.0081	0.4593±0.0777	0.4809±0.0323	0.4024±0.0125
ML-FBLS	0.0276±0.0006	0.0445±0.0027	0.4235±0.0071	0.544±0.0296	0.4925±0.0104
ML-RVFL	0.026±0.0005	0.0475±0.0033	0.3883±0.015	0.4644±0.0326	0.48±0.0154
ML-KRVFL	**0.024±0.0791**	**0.0393±0.0031**	**0.3151±0.0105**	0.3189±0.0143	**0.517±0.0051**

As described in Table 4.9, ML-KRVFL provides best hamming loss, ranking loss, one error, and average precision. ML-KRR/ML-KELM provides optimum coverage for rcv1v2s1 dataset.

Table 4.10: Comparison results performed on rcv1v2s2 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0365±0.0002	0.1652±0.003	0.7715±0.0109	0.3037±0.0587	0.2783±0.0156
ML-ELM[57]	0.0264±0.0004	0.1795±0.0165	0.5781±0.0377	0.6431±0.0217	0.4058±0.0069
BP-MLL[3]	0.0261±0.0002	0.6323±0.0278	0.7188±0.0772	**0.0772±0.0273**	0.0273±0.0146
ML-KRR/ ML-KELM[24]	0.0255±0.0003	0.4083±0.0102	0.802±0.049	0.1955±0.0097	0.4372±0.007
ML-BLS	0.0262±0.0006	0.097±0.0122	0.5354±0.0437	0.5299±0.0374	0.3714±0.0201
ML-FBLS	0.0254±0.0009	0.0754±0.0025	0.4567±0.0129	0.5735±0.0344	0.4524±0.0109
ML-RVFL	**0.0226±0.0005**	**0.0448±0.002**	0.3887±0.0144	0.4652±0.0333	0.386±0.0221
ML-KRVFL	0.0243±0.1081	0.0485±0.0133	**0.3465±0.0067**	0.5504±0.0114	**0.4766±0.0038**

As described in Table 4.10, ML-RVFL provides optimum hamming loss and ranking loss. ML-KRVFL provides optimum one error and average precision. BP-MLL provides optimum coverage for rcv1v2s2 dataset.

Table 4.11: Comparison results performed on rcv1v2s3 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0355±0.0025	0.1645±0.0049	0.7622±0.0103	0.304±0.0088	0.3077±0.0069
ML-ELM[57]	0.0259±0.0002	0.1753±0.0086	0.5809±0.0474	0.6224±0.0194	0.4147±0.0082
BP-MLL[3]	0.0259±0.0004	0.6246±0.0075	0.7267±0.0228	0.5392±0.0242	0.0893±0.0095
ML-KRR/ ML-KELM[24]	0.0254±0.0004	0.4251±0.0066	0.8178±0.0228	**0.1983±0.0078**	0.4398±0.0137
ML-BLS	0.0258±0.0007	0.0661±0.0099	0.5418±0.0568	0.3635±0.0334	0.3898±0.0181
ML-FBLS	0.0242±0.0007	0.0485±0.0019	0.4245±0.0121	0.6189±0.0205	0.4452±0.0154
ML-RVFL	0.0225±0.0007	0.0377±0.0014	0.2565±0.01	0.3963±0.0336	0.439±0.0167
ML-KRVFL	**0.017±0.0876**	**0.0209±0.0203**	**0.1516±0.0103**	0.2551±0.0231	**0.4856±0.0092**

For the rcv1v2s3 dataset ML-KRVFL provides optimum results for hamming loss, ranking loss, one error, and average precision. ML-KRR/ML-ELM provides optimum coverage. This is shown is Table 4.11.

Table 4.12: Comparison results performed on rcv1v2s4 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM [8]	0.0295±0.0005	0.141±0.0033	0.746±0.0144	0.2689±0.0044	0.3516±0.009
ML-ELM[57]	0.0236±0.0004	0.1827±0.0123	0.5679±0.0611	0.5941±0.023	0.4752±0.0137
BP-MLL[3]	0.0246±0.0003	0.5784±0.0254	0.804±0.053	0.5167±0.0588	0.0865±0.0187
ML-KRR/ ML-KELM[24]	0.0238±0.0003	0.4251±0.011	0.7663±0.0435	0.1701±0.0048	**0.4996±0.009**
ML-BLS	0.0295±0.0317	0.1657±0.11	0.5215±0.4528	0.3703±0.6085	0.4136±0.1575
ML-FBLS	0.0217±0.0013	0.11±0.0045	0.4528±0.0121	0.6085±0.028	0.4575±0.0096
ML-RVFL	0.0196±0.0003	**0.0358±0.0017**	**0.3262±0.0084**	0.1418±0.0157	0.4002±0.0189
ML-KRVFL	**0.0144±0.0912**	0.0396±0.0188	0.3314±0.0285	**0.1385±0.0236**	0.4847±0.0233

ML-KRVFL provides optimum hamming loss and coverage. ML-RVFL provides optimum ranking loss and one error. ML-KRR/ML-KELM provides optimum average precision. The experimental results on rcv1v2s4 dataset are mentioned in Table 4.12.

Table 4.13: Comparison results performed on rcv1v2s5 dataset

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.0361±0.0004	0.1574±0.0024	0.7528±0.0109	0.2993±0.0041	0.3113±0.0167
ML-ELM[57]	0.0259±0.0005	0.1853±0.0085	0.59±0.0526	0.6331±0.0237	0.4416±0.01
BP-MLL[3]	0.0262±0.0002	0.5892±0.0232	0.7941±0.0608	0.511±0.0202	0.1045±0.0125
ML-KRR/ ML-KELM[24]	0.0258±0.0003	0.4169±0.0059	0.8376±0.0325	**0.185±0.0039**	0.4567±0.0115
ML-BLS	0.0241±0.0008	0.0606±0.0053	0.4691±0.0452	0.3538±0.0175	0.4094±0.0084
ML-FBLS	0.0237±0.0006	0.0462±0.0023	0.4227±0.0116	0.4434±0.026	0.4643±0.024
ML-RVFL	0.0225±0.0007	0.0429±0.0008	0.3783±0.0113	0.2721±0.0259	0.44±0.0107
ML-KRVFL	**0.0221±0.0776**	**0.042±0.0123**	**0.2166±0.0146**	0.2137±0.0171	**0.4763±0.0044**

For the rcv1v2s5 dataset ML-KRVFL provides optimum hamming loss, one error, ranking loss, and average precision. The optimum result for coverage is provided by

ML-KRR/ML-KELM for rcv1v2s5 dataset.

Table 4.14: Average of performance measures on multi-label datasets

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	0.09525±0.00321	0.2202±0.0086	0.6842±0.5833	0.571525±0.0523	0.3728±0.0130
ML-ELM[57]	0.0791±0.010	0.4462±0.0221	0.4153±0.0625	0.6091±0.0756	0.4156±0.0388
BP-MLL[3]	0.0868±0.0031	0.408±0.0232	0.6890±0.0771	0.3495±0.021	0.3701±0.0136
ML-KRR/ ML-KELM[24]	0.1719±0.0394	0.3602±0.0133	0.6376±0.0542	**0.2534±0.0088**	0.3843±0.016
ML-BLS	0.0649±0.0060	0.1615±0.02145	0.4546±0.088	0.4778±0.0782	0.4088±0.0330
ML-FBLS	0.0583±0.0038	0.1089±0.0069	0.3787±0.01891	0.6601±0.0370	0.4524±0.0180
ML-RVFL	0.0558±0.0037	0.0751±0.0067	0.2931±0.0272	0.3826±0.0433	0.5127±0.0655
ML-KRVFL	**0.0532±0.0446**	**0.0714±0.0138**	**0.251±0.0180**	0.2620±0.0168	**0.5684±0.0126**

It can be observed from Table 4.2 to Table 4.13 that proposed multi-label approaches are performing better, specially ML-KRVFL and ML-RVFL are better in terms of most of the measures over all the twelve datasets. The size of datasets is important for the computation of the parameters and hyperparameters for the algorithms. At the time of calculation, the number of labels for the dataset is taken into consideration. This consideration affects the overall time of the training procedure. It is difficult for a single multi-label algorithm to perform better than all other algorithms for all five evaluation measures. For a better comparison, the average of each evaluation metric is taken on all datasets for algorithms used for comparison. In this case, Table 4.14 is used for the average of all evaluation metrics on all 12 multi-label datasets used for a fair comparison. This table is similar to Friedman's ranking [140] as the average is taken for evaluation metrics over the 12 multi-label datasets.

Table 4.15: Average Friedman rank for multi-label classification algorithms

	Hamming Loss	Ranking Loss	One Error	Coverage	Average Precision
RankSVM[8]	7.16	5.16	6.6	4.25	5.9
ML-ELM[57]	5.33	7	4.41	6.83	4.66
BP-MLL [3]	5.5	6.33	7	3.91	5.08
ML-KRR/ ML-KELM[24]	5	6.58	6	2.58	5.25
ML-BLS	5.41	4.5	4.41	4.91	5.75
ML-FBLS	3.16	2.91	3.33	7.16	3.66
ML-RVFL	**2.08**	1.91	2.5	3.83	3.83
ML-KRVFL	**2.08**	**1.58**	**1.66**	**2.5**	**1.83**

We follow the steps mentioned in [140, 141] and compute the Friedman rank of

each multi-label classifier based on each evaluation metric. Table 4.15 represents the average Friedman ranking for five evaluation measures over 12 multi-label datasets. The lower value of rank represents the higher performance in Table 4.15. As per the values mentioned in this table, ML-KRVFL, ML-RVFL, and ML-FBLS outperform other multi-label approaches. ML-BLS also has a better average Friedman ranking in comparison to other compared approaches, but ML-KRVFL, ML-RVFL, and ML-FBLS are better than all other approaches. The Friedman statistic for hamming loss, ranking loss, one error, coverage, and average precision is computed as 42.2, 64.16, 53.19, 42.5, and 25.5, respectively. The F-distribution with 11 and 77 degrees of freedom for hamming loss, ranking loss, one error, coverage, and average precision is computed as 11.10, 35.57, 18.99, 11.26, and 4.79. After performing the nemenyi post-hoc test [142] the critical difference for hamming loss, ranking loss, one error, coverage, and average precision is 3.031 on the basis of the significance level of 0.05.

Table 4.14 computes the average performance of individual approaches on twelve datasets from various domains and sizes. Further, Wilcoxon signed-rank test is conducted to verify the significance of performance between each pair of compared multi-label approaches. Wilcoxon signed-rank test considers the difference between the evaluation measures provided by the pair of classification approaches[133]. Tables 4.16, 4.17, 4.18, 4.19, and 4.20 denote the p-value obtained by conducting Wilcoxon signed-rank test on the basis of significance level 0.05 for five evaluation measures. The Wilcoxon signed-rank test is performed between each pair of compared approaches RankSVM, ML-ELM, BP-MLL, ML-KRR/ML-KELM, ML-RVFL, ML-KRVFL, ML-BLS, and ML-FBLS based on the evaluation measures hamming loss, ranking loss, one error, coverage, and average precision. The terms $--$ in tables 4.16, 4.17, 4.18, 4.19, and 4.20 denote that there is no self comparison is possible based on the Wilcoxon signed-rank test. The Wilcoxon signed-rank test performed based on evaluation measures indicates that RVFL based multi-label approaches provide significant results. ML-RVFL and ML-KRVFL are better than other approaches, and ML-FBLS provides better significant results in comparison to those approaches which are not based on RVFL.

Table 4.16: Statistics of p-values obtained from Wilcoxon's signed rank test for hamming loss

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
ML-RVFL	0.00222	0.00374	0.00288	0.00288	–	0.34722	0.00222	0.21498
ML-KRVFL	0.00288	0.00374	0.00328	0.0048	0.3472	–	0.0232	0.05
ML-BLS	0.00222	0.03318	0.5552	0.93624	0.00222	0.0232	–	0.00222
ML-FBLS	0.00374	0.0232	0.1878	0.9894	0.21498	0.05	0.00222	–

Table 4.17: Statistics of p-values obtained from Wilcoxon's signed rank test for ranking loss

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
ML-RVFL	0.00222	0.00222	0.00288	0.00222	–	0.38978	0.00222	0.0048
ML-KRVFL	0.00222	0.00222	0.002888	0.00222	0.38978	–	0.00374	0.01208
ML-BLS	0.30772	0.00222	0.0232	0.00288	0.00222	0.00374	–	0.01878
ML-FBLS	0.00222	0.00222	0.00374	0.00222	0.0048	0.01208	0.01878	–

Coverage is considered as the depth to cover all truth labels. From Table 2.6, it can be seen that the datasets having the number of output, the maximum number of labels is corel5k and the minimum number of labels are 6 in emotions and scene dataset. This measure also depends on the raw numerical values as an output produced by the different classifiers [143]. This is ensured that depth does not affect the finding of ground truth labels for the classifiers. The average hamming loss, one error, ranking loss, and average precision for proposed multi-label approaches are better than other competitive approaches in the majority of the evaluation metrics. The main assumption for these non-iterative approaches is the number of labels Q, which ranges from a minimum of 6 to 374 in the datasets. The difference among the labels becomes numerically less with the increase in the number of labels. This limitation exists for both iterative and non-iterative based types of solutions. In the future, The proposed multi-label approaches ML-RVFL, ML-KRVFL, ML-BLS, and ML-FBLS without threshold function can be mixed with semantic word embedding of labels. This approach is a better intuition to get better performance for larger numbers of labels. The computation of inverse matrix used in the non-iterative learning for parameter and hyperparameter computation can be inefficient for a large matrix.

Table 4.18: Statistics of p-values obtained from Wilcoxon's signed rank test for one error

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
ML-RVFL	0.00222	0.0232	0.00222	0.0048	–	0.0278	0.0048	0.151
ML-KRVFL	0.00222	0.0151	0.00222	0.0048	0.0278	–	0.00596	0.00758
ML-BLS	0.00758	0.93624	0.00222	0.01208	0.0048	0.00596	–	0.0232
ML-FBLS	0.00222	0.20766	0.00374	0.0096	0.151	0.00758	0.0232	–

Table 4.19: Statistics of p-values obtained from Wilcoxon's signed rank test for coverage

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
ML-RVFL	0.93624	0.01208	0.69654	0.04136	–	0.0096	0.11642	0.0222
ML-KRVFL	0.09894	0.0048	0.238	1	0.0096	–	0.04136	0.00222
ML-BLS	0.00222	0.13622	0.18352	0.00288	0.11642	0.04136	–	0.00222
ML-FBLS	0.034	0.93624	0.0048	0.0022	0.00222	0.0222	0.00222	–

Table 4.20: Statistics of p-values obtained from Wilcoxon's signed rank test for average precision

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
ML-RVFL	0.00758	0.38978	0.7186	0.278	–	0.0048	0.00596	0.4777
ML-KRVFL	0.00374	0.0278	0.232	0.00288	0.0048	–	0.00288	0.00596
ML-BLS	0.38978	0.477	0.5823	0.93624	0.00596	0.00288	–	0.00222
ML-FBLS	0.18352	0.5287	0.30772	0.0601	0.477	0.00596	0.00222	–

The kernelized-based approaches open the future scope for kernelized approaches in combination with fuzzy systems to improve all evaluation measures. The running time for the training process is mentioned in Table 4.21.

Table 4.21: Running time in seconds for training procedure

	RankSVM[8]	ML-ELM[57]	BP-MLL[3]	ML-KRR/ ML-KELM[24]	ML-RVFL	ML-KRVFL	ML-BLS	ML-FBLS
emotions	1096	0.1328	1021	0.41	0.14	0.38	0.4	0.42
yeast	27720	1.6929	25200	0.49	1.5	0.42	0.45	0.47
scene	543	4.25	508	0.06	4	0.08	0.03	0.04
corel5k	5974	19.65	542	38	20	36	30	34
enron	6216	42.66	1182	90	40	74	78	84
medical	8266	62.66	1468	126	64	115	114	118
bibtex	14568	98.62	11652	145	99	132	132	138
rcv1v2s1	19440	102	18420	114	96	106	98	104
rcv1v2s2	19480	104	18480	118	98	103	100	102
rcv1v2s3	19510	102	18390	116	99	101	102	108
rcv1v2s4	19420	103	18370	117	101	101	101	105
rcv1v2s5	19520	105	18470	118	103	102	99	106

The ML-KRVFL and ML-RVFL perform better for multi-label classification with

the limited tuning based on the five evaluation measures significantly. The performance of ML-BLS is improved by ML-FBLS using the fuzzy rules in it. Based on the Friedman rankings mentioned in Table 4.15 ML-KRVFL, ML-RVFL, and ML-FBLS provide optimum Friedman rankings for hamming loss, ranking loss, and one error measures. For the coverage, ML-KRVFL, ML-KRR/ML-KELM, and ML-RVFL provide optimum Friedman rankings, respectively. ML-KRVFL, ML-FBLS, and ML-RVFL provide optimum rankings respectively, as per the Friedman rankings. The overall performance of randomized neural networks for multi-label classification is significantly better than non-randomized-based approaches on the basis of experimental results.

4.5 Analysis of Computational Complexity

For the ML-RVFL, the computational complexity to compute the output of the enhancement layer is considered as $O(NLM)$. The computational complexity of computation of output weights for ML-RVFL is computed as the $O(L^3+L^2N+LNQ)$. The computational complexity of the threshold function is considered as $O(N^2Q)$. Hence the total computational complexity of ML-RVFL can be estimated as $O(NLM+L^3+L^2N+LNQ+N^2Q)$. The number of hidden neurons is denoted as L.

The computational complexity of computing kernel matrix for ML-KRVFL for linear kernel is $O(M)$ and for radial basis kernel function is $O(N^2M)$. The output weight for ML-KRVFL has the computational complexity $O(2N^3+QN^2)$. The threshold function has computational complexity $O(N^2Q)$. The complete complexity of ML-KRVFL is $O(M+N^2M+2N^3+QN^2+N^2Q)$.

4.6 Summary

In this chapter, we have proposed ML-RVFL and ML-KRVFL for multi-label classification. In these approaches, we have used pseudoinverse to compute the output weights. These approaches use an adaptive threshold function at the output layer

to find the set of associated labels for the test instances. Based on the Wilcoxon signed-rank test and Friedman test, the experimentation results on 12 benchmark multi-label datasets show that RVFL based multi-label approaches ML-RVFL and ML-KRVFL outperform the existing state-of-the-art multi-label algorithms for most of the evaluation metrics. The ML-RVFL and ML-KRVFL use the matrix inverse to compute the parameters of the neural network. The computation of inverse for large matrices is complex; due to this reason, it becomes cumbersome to compute the parameters for a large dataset for non-iterative approaches. We have proposed two non-iterative approaches, ML-BLS and ML-FBLS, in chapter 3 and further two non-iterative approaches, ML-RVFL and ML-KRVFL, in this chapter. All four non-iterative multi-label approaches ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL, need the preprocessed features of datasets and these approaches are unable to work with a large amount of raw data. In the next chapter, we have proposed a deep learning based multi-label classifier that is capable of overcoming these limitations.

Chapter 5

Graph Convolution Network based Multi-Label Classification

In this chapter, we have proposed a multi-label classifier for the image domain using a deep learning approach. The deep learning approaches are able to work on a large amount of raw data, which overcomes the shortcomings of non-iterative algorithms. We have proposed a graph convolution network based multi-label classifier for the image domain. The objects in an image usually have dependencies when the objects occur in an image. These dependencies are needed to consider during the multi-label classification. The proposed classifier multi-label graph convolution neural network with pairnorm ($MLGCN_{pairnorm}$) is able to work with a large number of labels. This approach captures the correlations among labels that provide the scalability to the number of labels. The proposed approach designs an inter-dependent classifier having the graph neural network with the convolution neural network. The $MLGCN_{pairnorm}$ addresses the over-smoothing problem that happens in GCN. The proposed classifier $MLGCN_{pairnorm}$ has two main components one is the image module, and the second is the GCN module. The image module extracts the features of the image, and the GCN module presents the labels as the nodes of graphs. These nodes are the semantic representation of the labels, which are usually a feature representation of labels. The proposed classifier is able to tackle the over-smoothing problem efficiently. Experiments performed on two large-size image datasets (MS-COCO and VOC 2007) show that the proposed approach alleviates the over-smoothing problem and performs

better than other multi-label classifiers. We use the benefits of graph data structure to explore and capture the label correlation and dependency. We utilized the GCN to consider the label co-occurrence and proposed an efficient end-to-end multi-label classifier that is able to handle the over-smoothing issues. The labels in the proposed approach are represented as nodes in the graph with their features (word embedding). The next section provides the introduction to multi-label classification in the image domain using GCN.

5.1 Introduction

Image recognition with multiple labels aims to find multiple objects in the given image. In this scenario, multi-label classification is different and more challenging than multi-class classification, where only one object is predicted in the given image [19, 20]. A simple approach for multi-label classification is to consider it as independent binary classifiers for each label. In these approaches, the label set grows exponentially and lacks the ability to capture semantic information of labels. Deep learning based approaches have great potential for computer-vision recognition and verification-related tasks. These approaches use the learned features from the input instances. CNN is extended for multi-label classification using various approaches, such as images directly fed into the CNN using the support vector machine pipeline[78]. Gong et al. [34] compared various losses using the CNN as a feature extractor. The top-k ranking provides the improvement in the performance of the multi-label classifiers [144]. All these mentioned approaches ignore the semantic relationship among labels. The semantic relationship between labels is being considered by the visual semantic embeddings in recent times [103, 145].

For the computer vision based tasks, a deep-learning based convolution neural network (CNN) can be trained with the softmax function at the output layer [79]. The softmax function at output layer based approaches also considers the multi-label classification as a different single-label classification problem, which uses the ranking loss or cross-entropy loss for the classification [34]. All of these approaches are limited

in their scalability as the number of classes or labels grows continuously. In the case of multi-label classification, as the number of classes increases, the distinction between classes becomes unclear. These models are unable to scale their ability for the growing number of labels or classes. The labels are dependent on each other in various manners, such as correlation, prior knowledge related to neighbor nodes, and geometric information of graphs. Among these dependencies, correlation is the key dependency for multi-label classification. The approaches discussed above fail to consider the correlations among labels.

The semantic label space can be designed by the information obtained from the unannotated text. This information preserves the semantic relationship among labels while visual-semantic embeddings are learned[103]. For the multi-label classification, RNN is utilized to consider the label dependencies among labels[146]. In this approach, the correlation among labels is learned by creating the link of label embeddings in the joint embedding space. A single CNN based paradigm for multi-label classification is proposed to learn from the transformation of an image rather than a representation of the image [145]. The CNN-based approaches are used to analyze the visual image data. This data represents the euclidean data, and CNN can not process the non-euclidean data (social network data), which is present in the real world in huge amounts [35]. The GCN is proposed to deal with this non-euclidean data. The GCN has two types of construction, spectral construction and spatial construction. The CNN can be explained in terms of spectral construction with the help of a mathematical foundation. The CNN can be used to design localized fast convolutions on graphs[147]. The process of convolution operators on graphs is improved by the step convolution operator and stacking the layer.

Thomas et al. [35] proposed the GCN, and later it was confirmed by Li et al. [84] that GCN is actually a variant of Laplacian smoothing. The GCN has the potential to capture the contextual information between the graph generation of labels and images. As shown in Figure 5.1, the probability of the presence of a cricketer in an image is high if bat, ball, stumps, and helmet occur together. The left image of bat and ball in Figure 5.1 helps to provide the semantic that if a ball is present, then the bat is

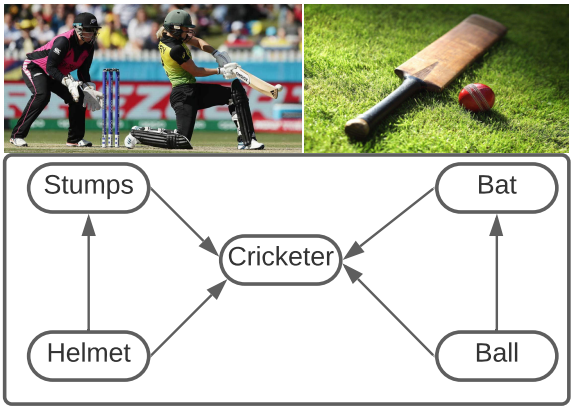


Figure 5.1: Example of label dependencies in multi-label classification

present in an image. These types of semantics help to create the graph of labels as the nodes and the dependencies as the edges. The edges in Figure 5.1 are mentioned to provide the illustration for the label dependencies. The dependencies are based on conditional probability. The dependencies are denoted by the direct graph. In this direct graph, we demonstrate a directed graph over the object labels to denote label dependencies in multi-label image recognition. For example, "Label A $\rightarrow$ Label B" means when Label A appears, Label B is likely to appear, but its reverse may not be true.

A graph convolution neural network based classifier is proposed to resolve the scalability and correlation issues [35, 148]. The graph convolution neural network based classifiers suffer from the over-smoothing problem [83, 84]. The over-smoothing problem occurs due to the increment in the number of layers. The convolution operators are analogous to laplacian smoothing. When the convolution operation is applied many times to the functions, then the functions converge to similar values. The same scenario happens when the layers are increased in the GCN then the features of nodes converge to similar values due to the convolution operations. This behavior is known as over-smoothing. The GCN suffers from over-smoothing problems because the nodes in the graphs are unable to differentiate among the neighbor nodes when the number of layers increases in the network. Over-smoothing makes it difficult to

make the GCN deep and decreases the performance of the deep classifier if it is not tackled in the initial phase of network design. In the next section, we present the proposed $MLGCN_{pairnorm}$ algorithm, which provides the solution to tackle the problem of over-smoothing in multi-label image recognition.

5.2 Multi-Label Classifier based on GCN to Tackle Over-smoothing Problem: $MLGCN_{pairnorm}$

In this section, we describe the proposed approach for multi-label classification with a graph convolution neural network based on pairnorm $MLGCN_{pairnorm}$ to alleviate the effect of over-smoothing. This approach is inspired by pairnorm, which keeps the total pairwise squared distance among the nodes similar before and after convolution operation [38]. The proposed approach incorporates the pairnorm to resolve the over-smoothing issue in the multi-label image domain. Lets $D = \{X, Y\}$ denoted the training set of data where $X = \mathcal{R}^{n \times d}$ denotes the training instance and $Y \in \{0, 1\}^{n \times d}$ denotes the labels. There can be more than one entry of 1 in Y is possible for the input instance. Given an image, we aim at partitioning labels into two disjoint sets according to the image-label relevance. In simple words, the proposed pairnorm based multi-label classifier separates the relevant and irrelevant labels of an input image using the graph convolution neural network.

We propose the multi-label classification approach to reduce the effect of over-smoothing using pairnorm with GCN. Pairnom can be considered as the normalization of the output of the graph convolution output. The graph convolution can be formulated as a graph regularized least square and considered as an optimization problem. The convolution procedure measures the variation of new features for the graph structure. Due to the convolution, the nodes with similar properties come in the same cluster, known as smoothing. The convolution procedure cannot distinguish the nodes from different clusters and performs the smoothing procedures on the nodes from different clusters. This smoothing process on distant nodes is termed as over-smoothing.

In the proposed approach, $\hat{A}$ represents the output of the graph convolution process, which is the input to the pairnorm. The output of the pairnorm process is denoted by $\dot{A}$. This flow is shown in Figure 5.2. In simple words, pairnorm is a normalization procedure that is applied after the convolution operation. This normalization process is a two-step process as follows.

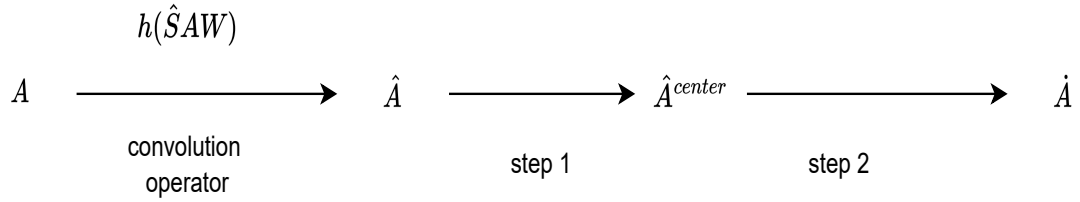


Figure 5.2: Flow diagram of pair norm

Step 1: In the first step, we subtract the row-wise mean from each $\hat{A}_i$ as follows

$$\hat{A}_i^{center} = \hat{A}_i - \frac{1}{n}\sum_{i=1}^{n}\hat{A}_i \tag{5.1}$$

where $\hat{A}_i^{center}$ denotes the centered representation of nodes and n is the total number of nodes in the graph which denotes the labels.

Step 2: Using the step two, scaling operation on centered representation is performed as follows.

$$\dot{A}_i = s_p\sqrt{n}.\frac{\hat{A}_i^{center}}{||\hat{A}_i^{center}||_F^2} \tag{5.2}$$

where $\dot{A}_i$ denoted the output of pairnorm and s_p is the scaling parameter. We have imposed more restrictions on node representation in the GCN and used the following form of scaling

$$\dot{A}_i = s_p.\frac{\hat{A}_i^{center}}{||\hat{A}_i^{center}||_2} \tag{5.3}$$

The total pairwise squared distance (TPSD) of nodes in matrix S and the output of convolution operation A^{l+1} becomes different, and the steps mentioned in Eq. 6.12 and Eq. 6.14 make it the same. The aim of pairnom is to keep the total pairwise squared distance among nodes constant before and after the convolution operation. These operations are performed after every convolution operation in the graph. In this

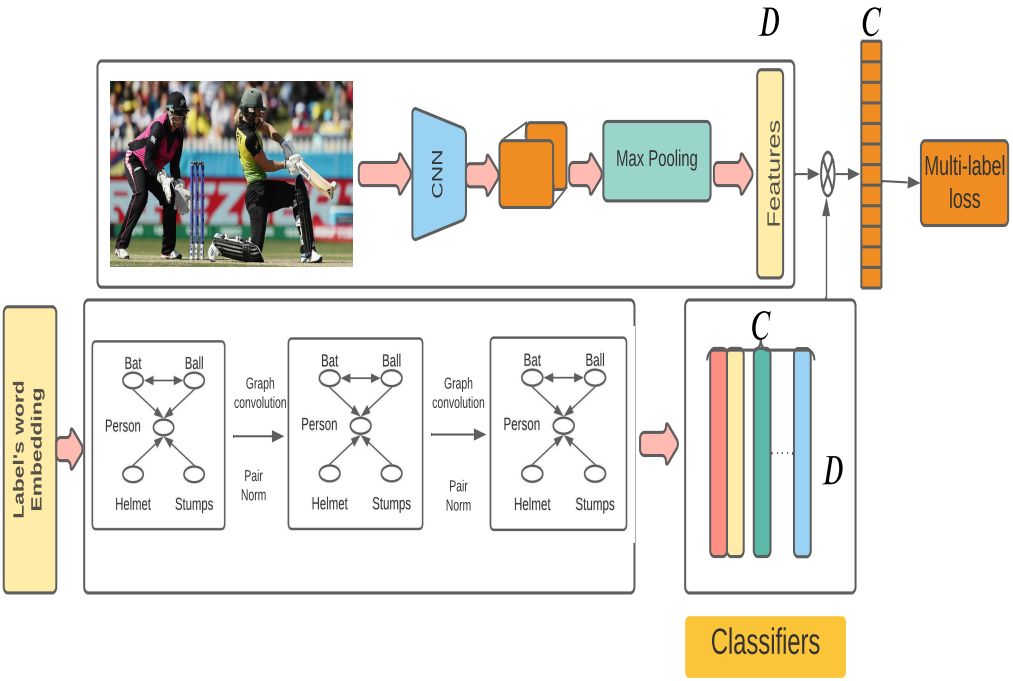


Figure 5.3: Proposed classifier with pair norm

way, the effect of over-smoothing is reduced in the GCN. In the GCN-based multi-label inter-dependent classifier, a matrix W is learned from the label representation using a GCN based mapping function. The GCN is used as stacking the layer l where the output of current layer A^l is processed by pairnorm, then it is provided to the next layer A^{l+1} as an input. The input for the first layer is $Z \in \mathcal{R}^{C\times d}$ matrix, and $\mathcal{W}$ are generated classifiers at the last layer having dimensions $\mathcal{R}^{C\times D}$. The predicted score of the classifier is obtained by applying the learned classifier to the image feature representation can be denoted as

$$\hat{y} = \mathcal{W}\mathbf{x} \tag{5.4}$$

The binary cross entropy (BCE) loss function is used to train the proposed model. The true labels are denoted by $y \in \mathcal{R}^{\mathcal{C}}$ and has the vaues 0 or 1 as per the association of labels. The BCE loss is defined below

$$\mathcal{L} = \sum_{c=1}^{C} y^c \log(\sigma(\hat{y}^c)) + (1 - y^c) \log(1 - \sigma(\hat{y}^c)) \tag{5.5}$$

Table 5.1: Parameter values of the classifier

Parameters	**Parameter values**
Hidden layer dimensions	1024, 2048
tau	0.4
LeakyReLU slope	0.2
Momentum	0.9
Weight decay	0.0001
Initial learning rate	0.01

As shown in Figure 5.3, the proposed approach has two parts. The first one is the image representation, and the second one is for the label's word embedding learning using GCN. For the first part, the image is converted to the features representation using the Rensnet101. Any CNN model can be used to generate feature representation. The generated features have the dimension same as the label's dimension D. In the second part, the embedding is learned using GCN, and C classifiers are generated. The dot product of each class is computed, and top labels are used for the prediction. The transformation from images to feature vectors can be done by CNN-based models. Any CNN-based model can be used as the backbone to transform the image into a feature vector. We use the ResNet-101 [31] as the backbone network to transform the image to the feature vector. After the transformation, the dimension of the image feature vector becomes equal to the dimension of the word embedding of labels. The dot product of word embedding and feature representation is computed, and the network is trained using BCE loss as shown in Eq. 5.5.

5.3 Experiments and Results

We discuss the experimentation settings and results reported by the proposed approach in this section. We have evaluated the performance of the proposed method on VOC 2007[5] and MS-COCO[20] datasets. We report the results based on the evaluation metrics mean average precision(mAP), per class average precision (CP), per class recall(CR), per class F1 score (CF1), and the average overall precision(OP), overall recall(OR), and overall F1(OF1) for the performance comparison.

The PyTorch framework is used to implement the proposed approach. We have reported the results on the two layers, three layers, and four layers. For the proposed ML-GCN, the results are reported to show the over-smoothing problem as the comparative parameter with the increasing number of layers. The labels are represented by the Glove [85], which is trained on the Wikipedia dataset. For those labels which have multiple similar words in embedding, we have taken an average of all the word features for similar words. We have taken the benefits of the mathematical properties of word embeddings for these labels. We can represent the labels in numerical features using embedding, so mathematical operations such as average, addition, and deletion can be used for the labels. LeakyReLU function is used for faster convergence in the image representation for nonlinear representation. The slope of LeakyReLU is negative for faster convergence. The ResNet architecture pretrained on the imagenet is used for feature extraction. Stochastic Gradient Descent (SGD) is used as an optimizer. The parameters of the network are shown in table 5.1. We first discuss the performance of the proposed approach on the MS-COCO dataset in section 5.3.1 and VOC 2007 in section 5.3.2. The ablation study for the over-smoothing is discussed in section 5.3.3.

5.3.1 Results on MS-COCO dataset

Microsoft-Common Objects in Context (MS-COCO) dataset is a widely used dataset for image classification tasks. This dataset contains the visual scenes. This dataset has been used for multi-label classification in recent times. There are 82,081 images present in the training set and 40,504 in the validation set. There are 80 classes in the MS-COCO dataset and an average of 2.9 objects per label. There are no specific labels available for multi-label learning for the test set, so multi-label approaches use the validation for testing purposes.

The experimental results for all the classes are reported in table 5.3. The mAP is significantly improved by the proposed classifier using pairnorm. The proposed pairnorm based $MLGCN_{pairnorm}$ outperforms all other classifiers for mAP CP, CR, CF1, OP, OR, and OF1 for all classes. The results for the top 3 classes are described in table 5.2. We compare the results with RNN-attention [82], CNN-RNN [80], ML-ZSL

Table 5.2: The comparison results for all labels on MS-COCO dataset

Methods	mAP	CP	CR	CF1	OP	OR	OF1
CNN-RNN	61.2	-	-	-	-	-	-
SRN	77.1	81.6	65.4	71.2	82.7	69.9	75.8
ResNET-101	77.3	80.2	66.7	72.8	83.9	70.8	76.8
Multi-Evidence	-	80.4	70.2	74.9	85.2	72.5	78.4
ML-CGCN	83.0	85.1	72.0	78.0	85.8	75.4	80.3
$MLGCN_{pairnorm}$	**85.114**	**87.86**	**75.54**	**81.23**	**89.66**	79.78	**84.43**

[94], Order free RNN [149], SRN [83] , Multi-evidence [150], and ML-CGCN [148]. The results show that the pairnorm scheme with ML-GCN performs better than other approaches for the top 3 classes also. It is observed that labels that are closer to each other become closer and distant labels remain distant from each other after applying the normalization using pairnorm. This is the reason that the performance of $MLGCN_{pairnorm}$ is significantly better than other state-of-the-art approaches.

5.3.2 Results on VOC 2007 dataset

The PASCAL VOC 2007 dataset is the collection of consumer photographs taken from the photo-sharing website Flickr. This dataset is well used for multi-label classification tasks. There are 9963 images from the 20 classes in the VOC dataset. We use the trainval set to train the model and the test set to test the proposed approach. The results of CP, CR, CF1, OP, OR, and OF1 for all classes are shown in Figure 5.4. The CP, CR, CF1, OP, OR, and OF1 are 91.63 %, 78.05 %, 84.29%, 98.03%, 96.14%, and 97.07 % respectively for all the classes of VOC 2007 dataset. The results of CP, CR, CF1, OP, OR, and OF1 for top 3 classes are shown in Figure 5.5. The CP, CR, CF1, OP, OR, and OF1 are 94.62 %, 76.58 %, 84.65%, 96.07%, 94.97%, and 95.51% respectively for top 3 classes of VOC 2007 dataset. For a fair comparison, we report the mean average precision (mAP) of the proposed approach with CNN-RNN[80], RLSD[151], VeryDeep[32], ResNet101[31], FeV+LV[152], HCP[146], RNN-Attention [82], Atten-Reinforce[153], and ML-CGCN[148] . The experimental results are shown in table 5.4 which shows that for the VOC 2007 dataset ML-GCN with pairnorm outperforms all other approaches.

Table 5.3: The comparison results for top 3 labels on MS-COCO dataset

Methods	CP	CR	CF1	OP	OR	OF1
CNN-RNN	66.0	55.6	60.4	69.2	66.4	67.8
RNN-Attention	79.1	58.7	67.4	84.0	63.0	72.0
Order-Free RNN	71.6	54.8	62.1	74.2	62.2	67.7
ML-ZSL	74.1	64.5	69.0	-	-	-
SRN	85.2	58.8	67.4	87.4	62.5	72.9
ResNET-101	84.1	59.4	69.7	89.1	62.8	73.6
Multi-Evidence	84.5	62.2	70.6	89.1	64.3	74.7
ML-CGCN	89.2	64.1	74.6	90.5	66.5	76.7
$MLGCN_{pairnorm}$	**91.55**	**66.83**	**77.26**	**94.21**	**70.04**	**80.35**

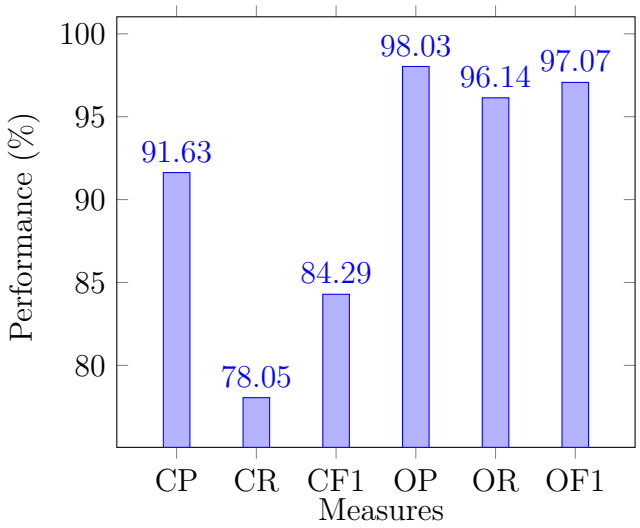


Figure 5.4: Results provided by the $MLGCN_{pairnorm}$ approach for all classes of VOC 2007 dataset

5.3.3 Ablation Studies

The over-smoothing problem is considered in the computer vision domain when the difference among classes becomes indistinguishable. The hidden layers in the network find the nonlinear transformation, and in the case of graphs, it provides the same value for all the nodes. Here nodes represent the labels. In this way, the labels become indistinguishable. The solution to the problem related to over smoothing is still being explored in recent times[84]. We propose a multi-label classifier based on a graph convolution neural network with an efficient method to minimize the effect of the over-smoothing problem. The result of increasing the layers in ML-GCN with

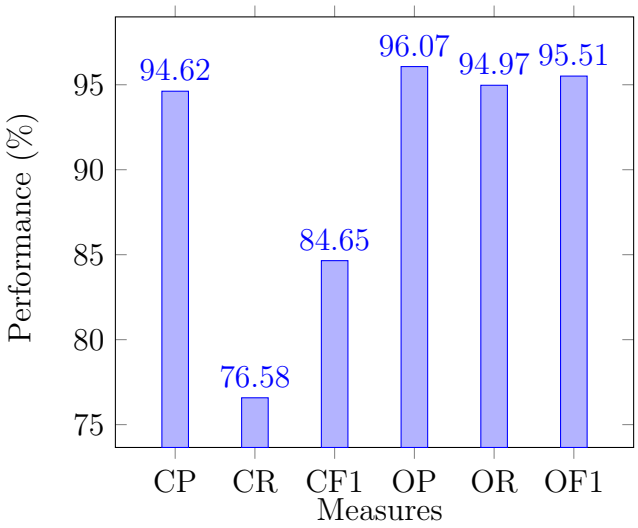


Figure 5.5: Results provided by the $MLGCN_{pairnorm}$ approach for top 3 classes of VOC 2007 dataset

Table 5.4: Comparison of mAP with state-of-the-art approaches on VOC 2007 dataset

Methods	**mAP**
CNN-RNN	84.0
RLSD	88.5
VeryDeep	89.7
ResNet101	89.9
FeV+LV	90.6
HCP	90.9
RNN-Attention	91.9
Atten-Reinforce	92.0
ML-CGCN	94.0
$MLGCN_{pairnorm}$	**94.47**

pairnorm is described in table 5.5 for MS-COCO dataset and in table 5.6 for VOC 2007 dataset. For the MS-COCO dataset, the effect of over-smoothing is minimized by having two layers. The pairnorm is applied to the two-layer GCN, and the value of mAP, CF1, and OF1 for all labels are 81.23%, 84.43%, 77.26%, respectively, and for the top 3 labels, CF1 and OF1 are reported as 80.34% and 75.67% respectively. For VOC 2007, the dataset mAP for two layers is reported as 94.47%, and for 3 and 4 networks, the reported mAP is 94.12% and 93.98%, respectively. These metrics are decreased when the number of layers is increased in the GCN network for the training of the classifier. It is observed that in the GCN-based approaches, the deeper networks

Table 5.5: The effect of over-smoothing on MS-COCO dataset

	All			top 3	
	mAP	CF1	OF1	CF1	OF1
2 layer	81.23	84.43	77.26	80.34	75.67
3 layer	80.00	84.12	76.98	80.12	75.67
4 layer	79.82	83.88	76.42	79.88	75.67

Table 5.6: The effect of over-smoothing on VOC 2007 datset

# Layers	mAP
2 layer	94.47
3 layer	94.12
4 layer	93.98

are not always better. Even in most of the scenarios, it suffers from the over-smoothing problem. So it's better to choose the appropriate approaches to minimize the effect of over-smoothing from the fewer layers itself so that the effect of over-smoothing can be alleviated for more layers. The pairnorm based interdependent classifier is able to alleviate over-smoothing for the two-layer network, and its performance becomes superior to other approaches when the layers are increased.

5.4 Summary

In this chapter, we have proposed $MLGCN_{pairnorm}$, which is GCN based multi-label classifier and able to process a large amount of raw image data. The $MLGCN_{pairnorm}$ is an effective inter-dependent object multi-label classifier which uses the feature extracted from the CNN and labels in the form of semantic embeddings. These semantic embeddings represent the features of the labels and are useful for creating the correlation matrix to consider the dependency among the labels. The $MLGCN_{pairnorm}$ incorporates the pairnorm for GCN to minimize the effect of over-smoothing. The pairnorm is easy to implement as it works as a normalization scheme for the output of graph layers. The GCN with pairnorm outperforms other state-of-the-art approaches for multi-label classification. The effectiveness of a pairnorm based multi-label classifier is shown by the experimental results performed on MS-

COCO and VOC 2007 datasets. The proposed approach is able to limit the effect of over-smoothing in multi-label classification. The aim of using this approach was to maintain the inter-dependence relationship between labels to alleviate the effect of over-smoothing. We only consider correlation as the inter-dependence relationship in this work which can be extended to use the geometry information and prior knowledge as an interdependence relationship among nodes. The $MLGCN_{pairnorm}$ have one main assumption that all the labels have their corresponding training instances during training. Apart from this assumption, some of the labels do not have their corresponding instances during training. The labels whose corresponding instances are available during training are known as seen labels and those labels whose corresponding instances are not available during training are known as unseen labels. The $MLGCN_{pairnorm}$ has a limitation in that it is unable to classify the instances which are corresponding to the unseen labels. In the next chapter, we have extended the $MLGCN_{pairnorm}$ to consider this transfer of knowledge from seen labels to unseen labels using zero-shot learning.

Chapter 6

Zero-Shot Learning for Multi-Label Classification using Graph Convolution Network

In this chapter, we have proposed a GCN-based zero-shot learning multi-label classifier for the image domain. The $MLGCN_{pairnorm}$ approach proposed in the previous chapter is unable to consider the unseen labels for multi-label classification. Hence we have extended the $MLGCN_{pairnorm}$ to consider the unseen labels using zero-shot learning. The proposed approach, multi-label zero-shot learning using pairnorm based partial label graph convolution neural network ($ML-ZSLPGCN$), uses the label features obtained from the images during training and semantic embeddings for the unseen labels. The $ML-ZSLPGCN$ first creates the features corresponding to the images, and its label aware module creates the feature vector of the labels corresponding to the seen labels. Zero-shot learning transfers the knowledge from the seen labels available during training to the unseen labels using semantic embeddings. A graph convolution network takes the feature vector of seen labels during training and semantic word embedding for the unseen labels as input and learns the classifier for multi-label classification. The next sections introduce the zero-shot learning in the image domain using GCN and discuss the proposed approach $ML-ZSLPGCN$.

6.1 Introduction

The deep learning algorithms for the multi-label classification using GCN are able to classify the raw image data. In the image domain, due to the various combinations of label spaces and mapping between image features and label space, ZSL multi-label spaces have the limitation of capturing the local features in the image and ignoring the global label dependencies [40]. In order to tackle these issues, Ou et al. have proposed the GCN based ZSL multi-label classifier in which the labels are represented as the nodes of the graph. This approach contains two modules [40]. The first one is the image module which captures the features of images, and the second is the GCN module which considers the semantic representation of label spaces. The semantic representation of labels is obtained by the pre-trained vectors such as Glove[85] and GoogleNews [87]. This GCN based approach uses a deep GCN network, and it suffers from the problem of over-smoothing[35, 84]. Due to this problem, these approaches are unable to differentiate the features of input data when the layers are increased in GCN. This situation is known as over-smoothing [84]. The over-smoothing problem occurs when the network becomes deeper in GCN.

To tackle these existing issues, we propose Multi-label zero-shot learning using pairnorm based partial label graph convolution neural network ($ML-ZSLPGCN$), which uses the label information from the images available during training and semantic embedding as well for unseen labels. The GCN uses the semantics embedding of the seen labels during training from the image, and for the unseen labels, it uses the word embedding of unseen labels. The proposed classifier is designed by using a simple normalizing pairnorm procedure [38] in the GCN to tackle the over-smoothing issue. $ML-ZSLPGCN$ can capture both class-specific features and label correlation from raw images, and it utilizes the information of image content and word embedding to learn an interdependent classifier. In the proposed approach, the features of labels for training data are generated using the label aware module, and semantic embedding is used for those labels which are not available during training. Both seen and unseen labels are used for training using GCN. The proposed approach takes the ben-

efits of seen labels from the label aware module and unseen labels from the semantic embedding. The main contributions of this chapter are mentioned below:

(i) Label aware module and semantic embedding both are used to represent the labels feature for seen and unseen labels, respectively, for multi-label zero-shot learning.

(ii) The over-smoothing issue in the zero-shot multi-label learning is tackled by introducing pairnorm.

6.2 Preliminaries

In this section, we present the preliminaries of multi-label zero-shot learning. Initially, the formal definition of zero-shot learning is described, followed by the discussion of Attention Regional Embedding, which is an important part of the proposed Multi-Label Zero-Shot Learning classifier.

6.2.1 Multi-Label Zero-Shot Learning

The aim of zero-shot learning multi-label classifiers is to provide good performance on both seen and unseen labels. Lets D^s represents the training set with instances X^s and their corresponding labels Y, where $X^s \in R^{n_s \times d}$ and $Y \in \{0,1\}^{n_s \times s}$. For the training set D^s, each training instance is represented by d dimensional vector. The test data D^t contains only the instances X^t which are represented by d dimensional vector. Let for the test data, instances related to both seen labels s and unseen labels u are available so we can represent the total number of labels as q using $q = u + s$, where set of unseen labels is denoted as $\mathcal{U} = \{s+1, s+2, ..., s+u\}$. For Zero-Shot learning, we assume that label embedding is available for both seen and unseen labels. In the proposed approach, the embedding of the seen labels is created using the label aware module, which is discussed in the next section. The embeddings of seen and unseen labels are represented by embedding matrix H=$[H_s : H_u] \in R^{q \times m}$ where $H_s \in R^{s \times m}$ represents the embedding matrix for seen labels and $H_u \in R^{u \times m}$

represents embedding matrix for unseen labels. The GCN generates correlated label representations of each label which is known as the interdependent classifiers C. These interdependent classifiers are used to classify the images. In the next subsection, we have discussed the attention regional embedding, which is used to find multiple objects automatically in the images.

6.2.2 Attention Regional Embedding

Attention Regional Embedding (ARE) captures discriminative regions without any manual annotation [154]. There are two parts of ARE; the first one is Attention Region Discovery(ARD), and the second is Attention Threshold(AT). ARD is used to highlight the attention regions, and AT filters out the regions having low attentive strength. After completing the operation of ARD and AT, the feature maps are concatenated by exploiting the Global Max Pooling(GMP). After these operations, a fully connected layer is used to control the dimension of the subnet and to fuse both local and global features to form the final image representation.

The discriminative regions in an image are captured automatically using the attention region embedding, which takes the features maps as an input provided by the backbone network. Any CNN network can be considered as the backbone network to extract the feature information from the image. The important regions in an image work like a bridge between an image x and its embedding for semantic transfer. These important regions can be captured using the attention mechanism. This attention mechanism is applied after the backbone convolution operation and obtaining the feature map Z from the convolution operation as shown in Figure 6.1. Suppose $Z \in \mathcal{R}^{height \times width \times channels}$ denotes the last feature map of the three-dimension tensor generated by the backbone network. This is generated by the CNN based backbone for image x as follows.

$$Z = \mathcal{B}(x) \tag{6.1}$$

where $B(x)$ denotes the final output of backbone network.

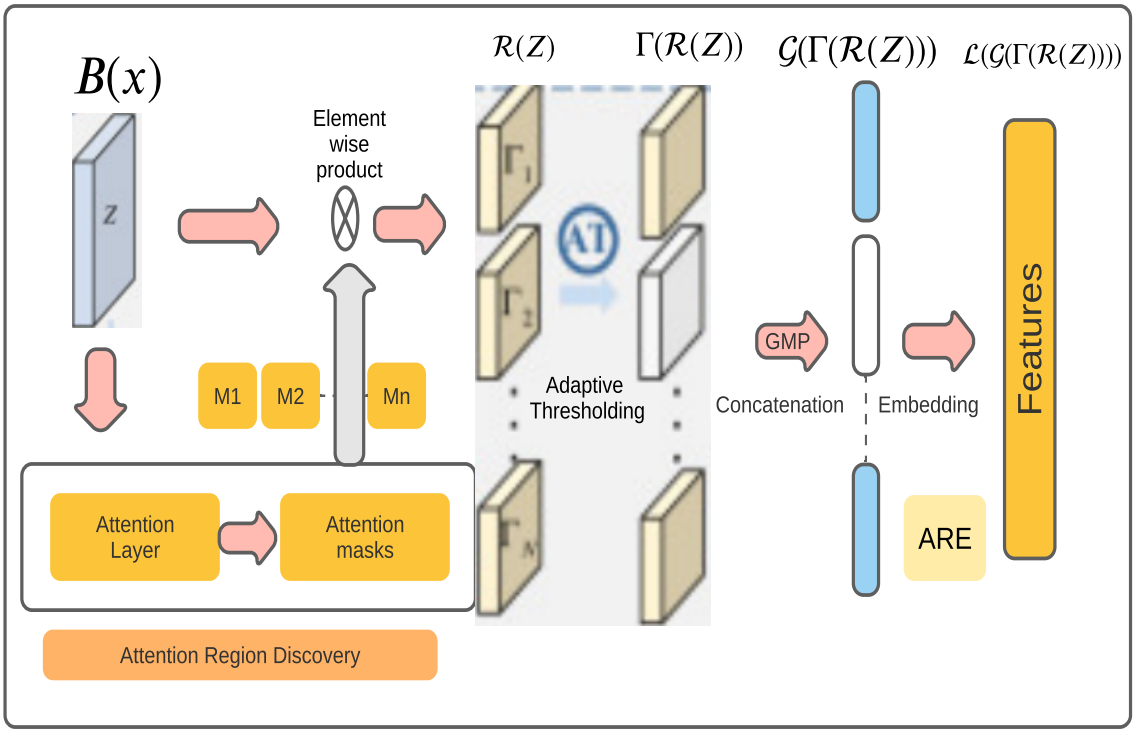


Figure 6.1: Attention Regional Embedding

Lets there are N no. of regions, and the attention mechanism captures the semantic region and reduces the gap between seen and unseen images. The following expression denotes the mask operation to create N number of two dimensions masks

$$M_n = \mathcal{M}_{MG_n}(Z) \tag{6.2}$$

where $\mathcal{M}_{MG_n}$ denotes the mask generation operation and $n \in (1, 2, ..., N)$. An element-wise product is performed between the generated mask M_n and the feature map Z to generate the attentions convolution feature maps. For this purpose, generated masks are reshaped with the size of Z.

$$\Gamma_n = \mathcal{O}_{Reshape}(M_n) \otimes Z \tag{6.3}$$

where Γ_n denotes generated attention convolution feature maps, $\otimes$ represents the element-wise product, and $\mathcal{O}_{Reshape}$, reshapes the size of the input to Z. These gen-

erated attentional convolution feature map contains redundancy, so adaptive thresholding (AT) operation is performed to filter the redundant noise. The AT operation first computes the maximum of each 2D mask map (M_n) from the N attention feature maps and generates the N dimensional maximum value vector $m_v \in \mathcal{R}^{N\times 1}$. The maximum value in m_v is known as global maximum value (AT_{max}) of N attention convolution feature maps. The following mathematical expression denotes the global maximum value of N attention convolution feature maps.

$$AT_{max} = \max_{1\leq n\leq N} m_v(n) \tag{6.4}$$

where AT_{max} represents the maximum value of vector $m_v(n)$ and n denotes the index of vector m_v. Hence $m_v(n)$ denotes the maximum value of each 2D mask map (M_n) from the N attention feature maps. To remove redundant noise, we use the threshold bound T_B which is denoted by the following expression.

$$T_B = \eta \times AT_{max} \tag{6.5}$$

where $\eta \in (0 \leq \eta \leq 1)$ is the adaptive coefficient, and T_B denotes the threshold bound which is used to generate the information-rich feature map by reducing redundancy. To generate the information-rich feature map, the value at n^{th} index in $m_v(n)$ is compared with T_B. If the value at n^{th} index in $m_v(n)$ is less than T_B then the corresponding feature map Γ_n is set to zero. In this way, the information-rich feature map is obtained. We use global max pooling to preserve the important features generated by ARD and concatenate them. The final output of the attention region module is a rich feature vector that is used in the proposed approach to generate the output layer and semantic feature vectors of labels in the label aware module.

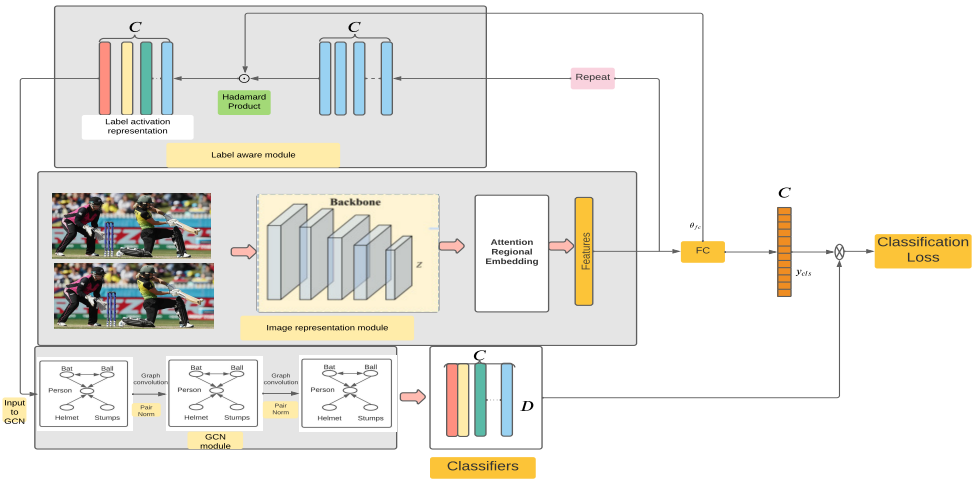


Figure 6.2: Complete $ML-ZSLPGCN$ approach using pairnorm

6.3 Proposed ZSL Multi-Label Classifier using GCN: $ML-ZSLPGCN$

In this section, we discuss the proposed GCN-based zero-shot learning multi-label classifier that contains three main modules. The first module represents the image representation module, the second module represents the label aware module, and the third module represents the learning module using GCN. The overall proposed approach is shown in Figure 6.2. We describe all modules one by one in subsequent sections.

6.3.1 Image representation module

In the image representation module, any CNN-based model can be used to learn the representation features of the input image x. In this work, ResNet101 is used as the backbone, which provides the feature maps of an image x to the Attention Regional Embedding (ARE) as shown in Figure 6.1. The semantic transfer of information between seen and unseen labels can be done using the local regions of an image corresponding to the specific semantic information. The fully connected layer is gen-

erated as the final image representation by concatenating the feature maps provided by the attention mechanism. The fully connected layer should match the dimension of the class/label. The extracted feature $f_v(x)$ is the final representation of image x, which is generated by the Image representation module. This representation can be represented using the following mathematical expression

$$f_v(x) = \mathcal{L}(\mathcal{G}(\mathcal{T}(\mathcal{R}(Z)))) \tag{6.6}$$

where $Z = \mathcal{B}(x)$, $\mathcal{R}$, $\mathcal{T}$, $\mathcal{G}$, and $\mathcal{L}$ are the backbone network operation, the ARD operation, the AT operation, the GMP operation, and the fully connected operation respectively. $f_v(x)$ is used in the label aware module to create the feature vectors for the seen labels, which are discussed in the next section.

6.3.2 Label-aware module

Label aware module is used to create the semantic embedding of those labels which are available during training. This module extracts the global image representation of an image x and converts it into a set of label-specific features. We use a fully connected layer to implement this module for x as shown in Figure 6.2. This can be represented as below.

$$y_{cls} = f_{fc}(f_v(x); \theta_{fc}), \tag{6.7}$$

where fc denotes the plain classifier, which forms the fully connected layer, θ_{fc} is the parameters of the plain classifier. Each classifier predicts the emergence of label c in the image on the basis of extracted information.

After the above procedure, the extracted label relevant features are obtained by using the following equation.

$$Z_{lar} = \mathcal{F} \bigodot \theta_{fc} \in \mathcal{R}^{C \times D} \tag{6.8}$$

where Z_{lar} denotes the relevant semantic feature vectors of seen labels and $\mathcal{F}$ denotes $[f_v(x), ..., f_v(x)]$, which represents C times copy of extracted features $f_v(x) \in$

$\mathcal{R}^D$. The $\bigodot$ symbol denotes the element-wise hadamard product. After applying the classifier parameters θ_{fc} on $\mathcal{F}$, the relevant semantic features vectors for seen labels are generated. Hence the hadmard product between $\mathcal{F}$ and classifiers parameters θ_{fc} captures the information related to the seen labels. The semantic label features Z_{lar} are used in GCN learning as an input for the seen labels.

6.3.3 Multi-Label Graph Convolution Network module

The Multi-Label Graph Convolution Network module uses GCN to represent the labels. In GCN, the labels are used as feature vectors and used for training. In the case of zero-shot learning, the feature vectors of seen labels are taken as an input which is taken from the label aware module. For the unseen labels, the label embeddings are provided as an input in GCN for the feature vector.

As discussed in section 2.4.1, the correlation matrix represents the cooccurance of labels for multi-label classification. In case of ZSL we propose two separate correlation matrices one for the seen labels and another for the transfer knowledge to unseen labels. For the seen labels during training, the binary correlation matrix S^1_{ij} is denoted as

$$\mathbf{S}^1_{ij} = \begin{cases} 0, \text{ if } \frac{n^{ij}}{n^i} < \tau \\ 1, \text{ if } \frac{n^{ij}}{n^i} \geq \tau \end{cases} \tag{6.9}$$

where $\frac{n^{ij}}{n^i}$ and τ are the probability and threshold respectively as discussed in section 2.4.1.

Using $\mathbf{S}^1_{ij}$ matrix, it is not possible to create the co-occurrence matrix between seen to unseen labels or unseen to unseen labels because unseen labels are not available during training. Hence we create the second similarity matrix for the transfer from seen to unseen labels as follows.

$$\mathbf{S}^2_{ij} = \begin{cases} 1, \text{ if } i \in \mathcal{N}_v(j) \text{ or j } \in \mathcal{N}_v(i) \\ 0, \text{ other} \end{cases} \tag{6.10}$$

where v nearest neighbors of label i are denoted by $\mathcal{N}_v(j)$. With the help of Eq. 6.9 and 6.10, complete correlation matrix is represented as follows

$$\mathbf{S}_{ij} = \begin{cases} \alpha \mathbf{S}^1_{ij} + (1-\alpha)\mathbf{S}^2_{ij}, \text{ if } i \in S \text{ and } j \in S \\ \mathbf{S}^2_{ij}, \text{ other} \end{cases} \tag{6.11}$$

where α can be used between 0 to 1. For our experimentation we have the value of α as 0.5 to provide the equal weightage to $\mathbf{S}^1_{ij}$ and $\mathbf{S}^2_{ij}$. Further we denote S_{ij} as S for notation simplification.

We design the final output of each GCN node to be the classifier of the corresponding label in our task. In the GCN module, the final output of each node is the classifier of the corresponding label. We create the label semantic similarity matrix using 300-dimensional embedding by calculating the Euclidean distance between labels. In this work, GCN takes the features generated by the label aware module for the seen labels and semantic embedding generated by the Glove embedding [85] for the unseen labels as an input. The GCN suffers from the problem of over-smoothing, so we incorporate the pairnorm to reduce the effect of over-smoothing. Pairnorm can be considered as the normalization of the output of the graph convolution output. The graph convolution can be formulated as a graph regularized least square and considered as an optimization problem. The convolution procedure measures the variation of new features for the graph structure. Due to the convolution, the nodes with similar properties are grouped in the same cluster, known as smoothing. The convolution procedure cannot distinguish the nodes from different clusters and performs the smoothing procedures on the nodes from different clusters. This smoothing process on distant nodes is termed over-smoothing [84].

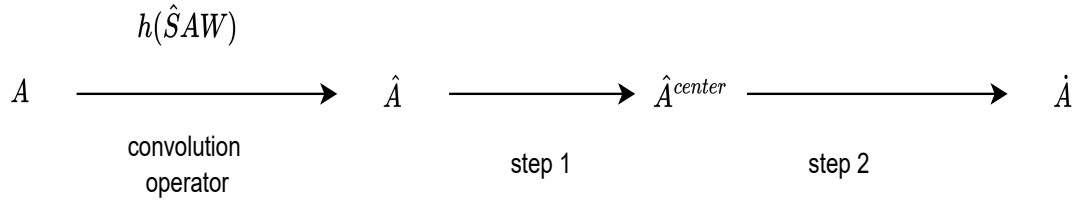


Figure 6.3: Flow diagram of pairnorm

The GCN takes the input features for seen and unseen labels as mentioned by the Eq. 2.6 and Eq. 2.7. In the proposed approach, $\hat{A}$ represents the output of the graph convolution process, which is the input to the pairnorm. The output of the pairnorm process is denoted by $\dot{A}$. This flow is shown in Figure 6.3. In simple words, pairnorm is a normalization procedure that is applied after the convolution operation. This normalization process is a two-step process as follows.

Step 1: In the first step, we subtract the row-wise mean from each $\hat{A}_i$ as follows

$$\hat{A}_i^{center} = \hat{A}_i - \frac{1}{n}\sum_{i=1}^{n}\hat{A}_i \tag{6.12}$$

where $\hat{A}_i^{center}$ denotes the centered representation of nodes and n is the total number of nodes in the graph which denotes the labels.

Step 2: Using the step two, scaling operation on centered representation is performed as follows.

$$\dot{A}_i = s_p\sqrt{n}.\frac{\hat{A}_i^{center}}{||\hat{A}_i^{center}||_F^2} \tag{6.13}$$

where $\dot{A}_i$ denoted the output of pairnorm and s_p is the scaling parameter. We have imposed more restrictions on node representation in the GCN and used the following form of scaling

$$\dot{A}_i = s_p.\frac{\hat{A}_i^{center}}{||\hat{A}_i^{center}||_2} \tag{6.14}$$

The total pairwise squared distance (TPSD) of nodes in matrix S and the output of convolution operation A^{l+1} becomes different, and the steps mentioned in Eq. 6.12 and Eq. 6.14 make it the same. The aim of pairnorm is to keep the total pairwise squared distance among nodes constant before and after the convolution operation. These operations are performed after every convolution operation in the graph. In this way, the effect of over-smoothing is reduced in the GCN. In the GCN-based multi-label inter-dependent classifier, a matrix W is learned from the label representation using a GCN based mapping function. The GCN is used as stacking the layer l where the output of current layer A^l is processed by pairnorm, then it is provided to the next layer A^{l+1} as an input. The input for the first layer is $Z \in \mathcal{R}^{C\times d}$ matrix, and $\mathcal{W}$

are the output generated classifiers of the last layer having dimensions $\mathcal{R}^{C\times D}$. The predicted score of the classifier is obtained by applying the learned classifier to the image feature representation can be denoted as

$$\hat{y} = \mathcal{W}\mathbf{y_{cls}} \tag{6.15}$$

After the discussion of the three modules, the proposed $ML-ZSLPGCN$ approach can be considered as a supervised model with all the seen and unseen labels. Due to ZSL supervised learning, we use quasi-fully supervised learning loss [155] during training which is defined as follows.

$$L = \frac{1}{n_s}\sum_{i=1}^{n_s} L_p(x_i^s, y_i) + \frac{\lambda}{n_t} L_b(x_i^t) \tag{6.16}$$

where λ is a parameter to provide the trade-off between two terms L_p and L_b, n_s is the number of training instances, and n_t is the number of test instances. The term $L_p(x_i^s, y_i)$ is the binary cross entropy loss defined as follows

$$L_p(x_i^s, y_i) = \sum_{c\in S} y_i^c log(\sigma(\hat{y}_i^c)) + (1-y_i^c) log(\sigma(1-\hat{y}_i^c)) \tag{6.17}$$

where $\hat{y}_i^c$ denotes the predicted output of c^{th} label for i^{th} instance and σ denotes the activation function. In Eq. 6.16, $L_b(x_i^t)$ is the balance term which is used to reduce the bias to the seen classes on the unlabelled data. The balance term can be denoted by the following mathematical expression.

$$L_b(x_i^t) = -ln\frac{\sum_{c\in S}(\sigma(\hat{y}_i^c))}{\sum_{c\in S}(\sigma(\hat{y}_i^c)) + \sum_{c\in \mathcal{U}}(\sigma(\hat{y}_i^c))} \tag{6.18}$$

where S denotes the seen labels set, and $\mathcal{U}$ denotes the unseen label set. The loss mentioned in Eq. 6.16 is optimized in the proposed $ML-ZSLPGCN$ model during training. The Image representation module is used to create the features, and the label aware module creates the feature vectors for the seen labels. The GCN is used to map the features of seen labels Z_{lar} and semantic embedding of unseen labels

Table 6.1: Dataset description

Dataset	NUS-WIDE	MS-COCO
training	1,00,000	78,081
validation	10,203	4,000
testing	20,000	40,137
classes	1,000/81	80

into the generated classifiers. The GCN maintains the inter-class relationship using the correlation matrix. The over-smoothing problem in GCN is taken care of by proposing pairnorm. The top associated labels are created using the dot product between generated classifiers and the output of the image representation module.

6.4 Experiments and Results

The experimental details of the proposed approach $ML-ZSLPGCN$ are discussed in this section. The experiments of the proposed $ML-ZSLPGCN$ are performed on XUbuntu 18.04 Intel Xeon Gold server with 192 GB RAM and TITAN Xp 16 GB GPU. The two benchmark datasets, MS-COCO and NUS-WIDE, are used to test the proposed pairnorm based model for zero-shot learning. The details of these two datasets are mentioned in Table 6.1. The details of the datasets NUS-WIDE [131] and MS-COCO [77] is followed from [148, 94] for the experimentation. Based on the K highest-ranked labels for the corresponding images, the evaluation measures precision, recall, and F1 score are computed to measure the performance of the proposed model. For the training, labeled and unlabelled data are mixed, and randomly selected images are used to create batches of size 32.

For the image representation, the input images are cropped to the size 448×448 with a random horizontal flip during training. The Resnet 101 is used to create the $2048 \times 14 \times 14$ feature maps. The proposed architecture uses a two-layer GCN-based classifier with the output dimension of 1024 and 2048. The Glove semantic embedding is used for the label representation. Labels having multiple words are embedded by averaging the features of an individual word in those labels. As the same semantic

Table 6.2: Parameters used in the proposed approach $ML - ZSLPGCN$

Parameters	Value
Initial learning rate	0.1
No. of Epochs	50- 200
Momentum	0.9
Weight decay	0.00001
Number of attention regions	10
AT parameter range	0.5 to 1
α	0.1 to 1

Table 6.3: Comparison results on MS-COCO dataset

	P	**R**	**F1**
WSABIE	54	55.7	54.8
Logistics	67.3	60.1	63.5
ML-KNN	54.9	56.7	55.8
Fast0Tag	57.2	61.3	59.2
MZSL-KG	72.8	63.4	67.8
ML-GCN	82.4	69.7	75.5
MZSL-GCN	83.7	71.3	77
ML-ZSLPGCN	**84.5**	**73.2**	**78.45**

embedding Glove is used for both datasets, The threshold τ for both datasets is set as 0.4 in the experimentation. The parameters of the network are mentioned in Table 6.2. The network is implemented on the Pytorch framework.

The proposed approach is compared with WSABIE [90], Fast0tag [156], Logistic [157], ML-KNN[25], MZSL-KG [94], ML-GCN [148], ESZSL [108], LFRLS [158], and MZSL-GCN [40]. The results of existing approaches are taken from the GCN based zero shot learning for multi-label classification [40].

Table 6.3 gives the details of experimental results obtained for the MS-COCO dataset. Our proposed pairnorm based $ML - ZSLPGCN$ approach provides superior results against the baselines approaches WSABIE, Logistics, ML-KNN, Fast0Tag, MZSL-KG, ML-GCN, and MZSL-GCN.

Table 6.4: Comparison results for NUS81 dataset

	P	**R**	**F1**
WSABIE	27	46.1	34.1
Logistics	35.2	43.3	38.8
ML-KNN	25	43.6	31.8
Fast0Tag	28.9	**51.2**	36.9
MZSL-KG	41.2	45.3	43.2
ML-GCN	53.1	37.3	43.8
MZSL-GCN	53.3	38.4	44.6
ML-ZSLPGCN	**53.6**	40.2	**45.94**

Table 6.4 gives the details of experimental results obtained for the NUS-81 dataset. $ML-ZSLPGCN$ provides superior results against the compared WSABIE, Logistics, ML-KNN, Fast0Tag, MZSL-KG, ML-GCN, and MZSL-GCN in terms of F1 score.

The value of α mentioned in Eq. 6.11 represents the effect of the unseen and seen labels correlation on matrix A. The 0 value of α represents correlation matrix A_{cs}^2 as A_{cs}. The 1 value of α represents the correlation matrix A_{cs}^1 as A_{cs}. The value of α ranges between 0 and 1, and the value toward 1 makes the correlation matrix suitable for seen labels learning, and The value of α towards zero makes the correlations matrix suitable for unseen labels. The balance of correlation between seen and unseen labels is needed for better multi-label zero-shot learning. In this work, the value is taken as 0.5, which provides equal weightage for seen and unseen labels correlation matrix.

The parameter λ mentioned in Eq. 6.16 has a significant effect on the performance of the multi-label ZSL performance. This parameter provides the balance to the loss function for seen and unseen labels. The small value of λ leaves the ZSL problem unsolved because the part of unseen labels becomes very less significant in the loss function according to the Eq. 6.16. The large value of λ provides weightage to the unseen label learning. The range of λ is considered between $[0, 0.1, ...0.9, 1]$.

The threshold variable τ converts the continuous values of the correlation matrix A to the binary matrix. The higher value of τ increases the number of filtering edges, due to which the F1 score gradually decreases. The 0 value of τ does not filter out edges, so it does not converge the model. The effect of τ on the F1 score is shown in

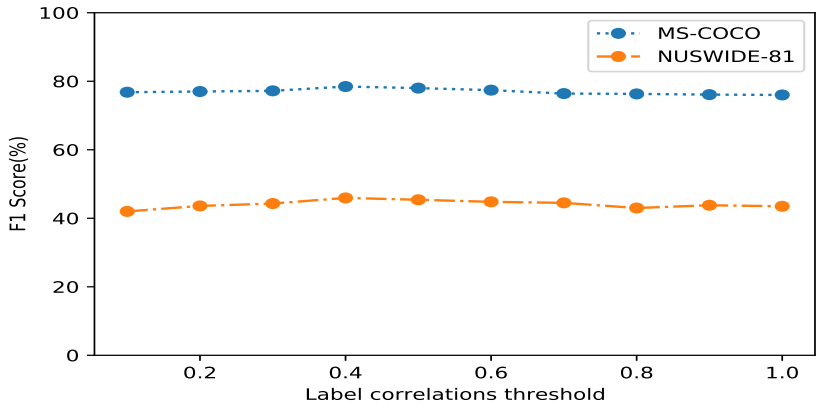


Figure 6.4: F1 score performance vs. label correlation threshold τ

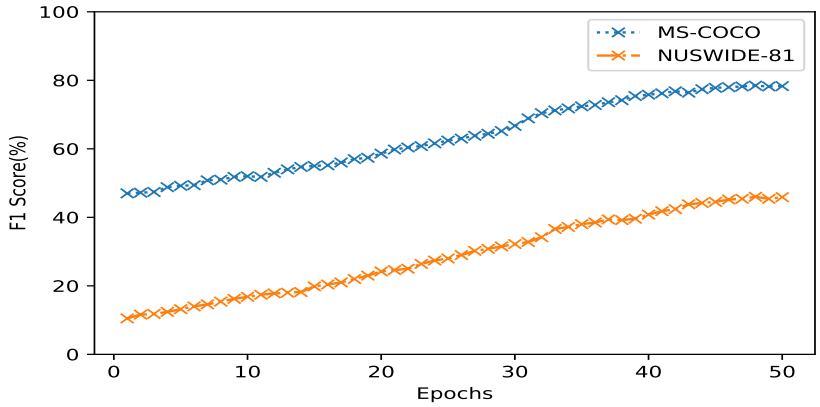


Figure 6.5: F1 score vs number of epochs

Figure 6.4. In this figure, both the datasets NUSWIDE-81 and MS-COCO have the highest F1 score on the 0.4 value of label correlation threshold τ. The value of τ is kept the same for all the datasets because it is based on the Glove embeddings of the nodes, which is similar for all the datasets. The graph shown in Figure 6.5 describes the improvement of the F1 score provided by the proposed $ML-ZSLPGCN$, which is based on the number of epochs for both MS-COCO and NUSWIDE-81 datasets.

6.4.1 Ablation Study

The ablation study to analyze the effect of the proposed pairnorm based $ML-ZSLPGCN$ is discussed in this section. Table 6.5 represents the precision, recall, and

Table 6.5: Ablation study for $ML-ZSLPGCN$

Layer		2-layer	3-layer	4-layer
MS-COCO	P	84.5	84.2	83.8
	R	73.2	72.8	72.2
	F1	78.445149	78.0861146	77.5687179
NUS-81	P	53.6	53.2	52.8
	R	40.2	39.8	39.6
	F1	45.9428571	45.5346237	45.2571429

F1 score by using 2, 3, and 4 layers in $ML-ZSLPGCN$. The performance of the GCN based approaches decreases by increasing the number of layers. This decrement can be tackled by using the pairnorm with the GCN based approaches so that the performance deduction can be minimized. The results on the datasets MS-COCO and NUS-81 are nearby similar by increasing the layers in $ML-ZSLPGCN$. The output dimensions for the three layers are set in the experimentation as 1024, 1024, and 2048. The output dimensions for four layers are set in the experimentation as 1024, 1024, and 2048.

6.5 Summary

In this chapter, we have proposed a GCN-based zero-shot multi-label classifier ($ML-ZSLPGCN$) that is able to alleviate the over-smoothing problem in GCN using pairnorm technique. The $ML-ZSLPGCN$ contains three modules. The first is the image representation module, which creates the features of images, and the second module is the label aware module which creates the features of seen labels. The third GCN module maps the class-relevant representations to the inter-dependence representation of classes. The co-occurrence of labels in GCN is captured by a correlation matrix which is created by two different types of correlation matrices. The first correlation matrix considers the co-occurrence of seen labels, and the second correlation matrix is used for transferring knowledge from seen labels to unseen labels. The experimentation results performed on MS-COCO and NUS-WIDE benchmark datasets show that $ML-ZSLPGCN$ is better than state-of-the-art multi-label approaches

in terms of precision, recall, and F1-score. Hence GCN based multi-label algorithms, $MLGCN_{pairnorm}$ and $ML-ZSLPGCN$ are able to classify the raw images with a large number of labels. However, these approaches are unable to classify the sequential data. Therefore, we have proposed the heuristic-based deep learning approach that is able to classify the sequential data and simultaneously handle the data imbalance problem.

Chapter 7

Heuristic based Deep Learning Multi-Label Classifier for Sequential Protein Data

In this chapter, we have proposed a heuristic-based deep learning algorithm for multi-label sequential data. The greatest challenge with multi-label classification is to handle the data imbalance, which appears due to variance in frequencies of the labels in the data. We have proposed a heuristic-based deep learning algorithm for functional annotation of primary protein structure, which is an imbalanced sequential data. The amino acids in protein sequences are responsible for multiple functions; hence the classification of protein sequences is multi-label in nature. The proposed approach improves the performance in comparison to state-of-the-art algorithms in terms of precision, recall, AUC, subset accuracy, and F1 score. We have proposed a heuristic approach to improve the precision and recall of the individual functional classes to handle the data imbalance problem. In the next section, we have introduced the multi-label classification of the protein sequences.

7.1 Introduction

In this section, we have introduced the multi-label classification for functional annotation of sequential protein structure. In [159], the authors propose a protein

classification approach in which proteins are stored in 20 x 20 bi-peptide matrices. The protein classification approach uses unsupervised learning. As a result of this, the self-organization of the neuron activation is used in a topologically ordered map, such that the proteins belonging to a known family are associated with the same neuron or one neighboring it. This self-organization into topologically ordered maps makes the classification fast for new inputs. This method filters 1758 protein sequences with lengths greater than 50. Another approach with the n-gram hashing or singular value decomposition(SVD) method is useful for applying protein sequences to encode the input vectors to the neurons [160]. This approach uses the annotated Protein Identification Resource (PIR) database, and the input is applied to a three-layered feed-forward neural network that employs the backpropagation learning algorithm. In this work, target classes are pairwise disjoint. Hence this solution does not consider the multi-label characteristic of protein sequences.

There are some approaches based on the Protein-Protein Interaction (PPI) datasets that use random walk [161] and fruit fly optimization-based approaches [162] to characterize the protein. The random walk based approach uses biological and topological properties to determine protein essentiality in PPI. This approach is based on static PPI networks and can not reflect the transient behavior of protein sequences. The fruit fly optimization-based approach is able to determine the dynamic behavior of protein, but this approach is unable to use the primary structure of protein sequences. Even though experimentally validated PPI data drives research and development of proteomics, they often have high false positives and false negatives. The cluster information and graph creation process is also time-consuming for the large available protein data. For the sequential data, RNN has been introduced to fetch the complex features and patterns[163]. The RNN still requires computation cost, and its two variants, Long Short Term Memory (LSTM) and Gated recurrent unit (GRU), still are unable to be in a parallel manner. These work in a sequential manner and the tasks using RNN can not be parallelized. Also, the RNN based models suffer from the vanishing or exploding gradient problem, which occurs based on the depth of the model. In [164], the neural network trained on 80% of the sequences of the SwissProt subset of

the UniProt Dataset and tested its performance on the remaining 20%. This provides a nearly 100% accuracy and classifies the proteins into only four different classes. In this work, filtering of sequence length was limited between 10-1000 or 10-2000 as per the classes. The 1-dimensional convolution neural network (1D-CNN) has the ability to find the receptive fields which overcomes the vanishing gradient problem. Hence we have proposed the 1-D CNN to classify the UniProt data which has more than a thousand classes.

In order to overcome the vanishing gradient problem, we have proposed a deep 1D-CNN to annotate the protein sequences on the basis of functional classes. This is our motivation to use 1D-CNN for the proposed heuristic based deep learning approach. We propose a 1D-CNN based heuristic approach to annotate the protein sequences using the available structure information, i.e., the primary structure, which contains the sequence of amino acids of a protein. We have filtered the UniProt data sequences to make them available to process with neural networks. The proposed heuristic approach is inspired by the harmonic heuristics [41] which has been adapted in various domains such as feature selection, clustering-based approaches, and optimization [165, 166, 167]. We have proposed the heuristic approach to improve the individual precision and recall of the labels which do not have better values by exchanging the weights in the loss function of the 1D-CNN. The heuristics improve the performance of the 1D-CNN based architecture. The raw primary structure of protein needs to be filtered so that it can be provided to the deep neural network as an input. The UniProt data and its filtering process is discussed in the next section.

7.2 UniProt Data Representation and its Filtering

In this section, we have discussed the representation of the sequential UniProt data, which has been used in the proposed approach, followed by the discussion of the embedding of protein sequences.

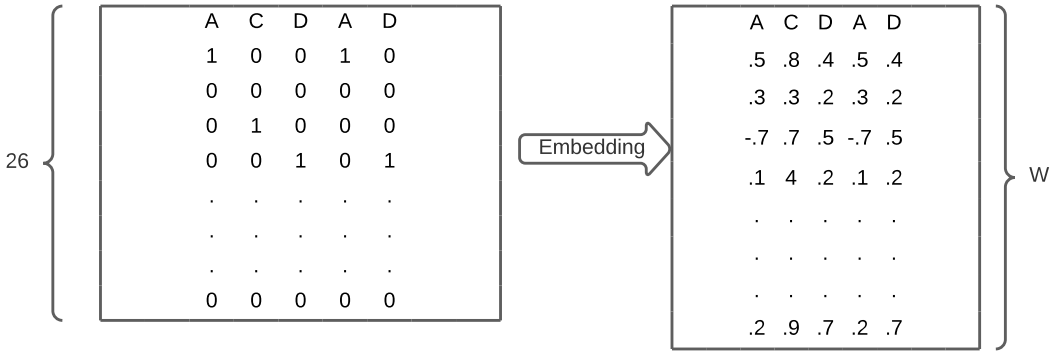


Figure 7.1: Embedding of ACDAD partial proteing sequence with size $W \times L$

7.2.1 Filtering of UniProt data

The UniProt database, which is used in the experimentation, is a primary sequential structure of the protein. After obtaining the raw data from the UniProt source, It is necessary to filter this data suitably. Data is shuffled and split into train and test data, with the test data containing 5000 entries and the rest of the data being used for training purposes. Some filtering measures are applied to make them suitable for learning algorithms. The filtering criteria for the train data are as follows.

(i) Constraining the sequence length between 162 and 2000. This is largely determined by the available video memory on our GPU(Nvidia GP102 TITAN Xp 12 GB GPU). The lower limit is set to 162 so that the output of the last pooling layer is at least one amino acid.

(ii) The starting character "M," which represents Methionine, is removed from all the sequences.

(iii) Once protein sequences are filtered, the corresponding labels are filtered such that every label has at least one protein sequence belonging to it.

For the testing purpose, we have considered the sequences with a length greater than 162. The longer sequences are cropped to a maximum length of 2000 for testing purposes.

7.2.2 Encoding of protein sequences

The primary protein sequences are the amino acid sequence where each amino acid is represented by the 20 characters mentioned in table 2.5. These sequences are converted into numerical representations using one hot-encoding. Hence the characters of protein sequences are converted into numerical values. Each character is converted to the 1-hot vector. We have used 20 amino acids and their six properties, so each character becomes the vector of 26 lengths. Suppose the length of the sequence is L, so each sequence is converted to the $26 \times L$. The 1-hot encoding is the vector of values 0 and 1, so these vectors are converted to the vector of continuous values with fixed length size W. The embedding is done by the *wordtovec* model named as ProtVec embedding. Each protein sequence becomes the size of $W \times L$. The embedding process for the partial sequence $ACDAD$ is shown in Figure 7.1. This embedding is fed to the proposed heuristic based deep learning approach, which is discussed in the next section.

7.3 Heuristic based Deep Learning approach for Multi-Label Protein Sequences

In this section, We discuss the proposed heuristic based 1D-CNN deep learning approach for the classification of primary protein structures. The proposed approach consists of an encoding module, deep learning based 1D-CNN module, and a heuristic module. In the first module, the protein sequences of amino acids and their six properties create the 26-dimensional vector for each amino acid. These sequences are converted to the fixed $W \times L$ size matrix using the embedding, and these sequences are fed to the 1D-CNN module followed by fully connected layers. The heuristic rules have been applied to the two generated models based on the best subset accuracy. The heuristic approach increases the precision, recall, and F1 score using the increase in the individual precision and recall of the labels. As shown in Figure 7.2 first, the protein sequences are converted into the numerical matrix to make them suitable for the

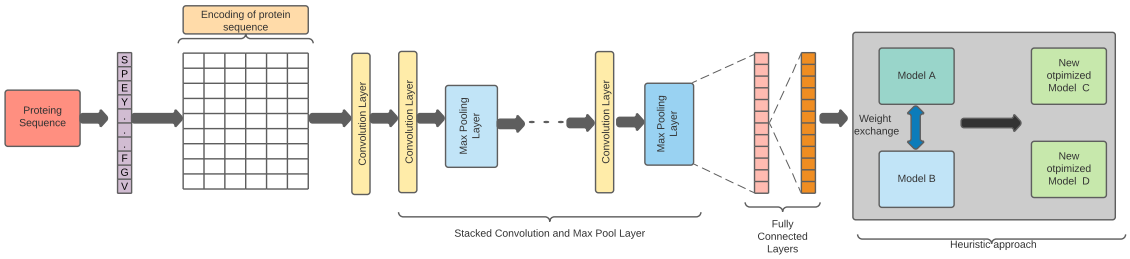


Figure 7.2: Heuristic based system architecture

convolution process. These matrices are passed by the different convolution and max pool layers. Nonlinear features of protein sequences are captured by these convolution layers. These protein sequences are considered analogous to text sequences.

We use a deep network of 1D-CNN for the classification of protein sequences. The stacking of convolution layers, max-pooling layers, and fully connected layers is used for the classification. At the end of fully connected layers, the heuristic approach is applied to improve the classification performance. We discuss the encoding of sequences and the heuristic based 1D-CNN approach in the next subsections.

The 1-D convolution takes into consideration the type of amino acid as well as its properties, which determine the protein structure while doing the convolution operation. The effect produced by certain constituent amino acids only lasts in its neighborhood and is not observed beyond it. The choice of the kernel size is used to restrict the learning only to the neighborhood. The information extracted by convolution operation effectively calculates the aggregated effect of groups of amino acids. With each pass through a convolution layer, the length of the sequence decreases depending upon the kernel size and stride. Therefore, in this process, complex information on neighborhood influence can be learned. After the convolution and pooling layers, the information is passed on to the fully connected layers of the neural network. At the final fully connected layer, the values are compared to the target vector, and the model is trained using the following loss function.

$$L(y, \widehat{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} -y_{ij} \log \widehat{y}_{ij} - (1 - y_{ij}) \log(1 - \widehat{y}_{ij}) \tag{7.1}$$

where y be the target output vector and $\widehat{y}$ be the output vector predicted by the model. The total number of samples are m and total number of labels are n.

The loss function mentioned in Eq. 7.1 penalizes the training algorithm if the output is not matched with the target vector. In the genome-related multi-label classification tasks, the frequency of the label is calculated for the entire data set. The distribution of labels is usually skewed for genome-related data. The learning task is difficult for the labels with less frequency in the dataset. The solution to this problem is to do over-sampling of labels with lower frequency or under-sampling of labels with higher frequency or adding weights in the loss function. The annotation task of protein sequences is multi-label in nature, so over-sampling a label occurring a few times can also lead to over-sampling of a frequently occurring label. A similar case can be considered for the under-sampling of labels. Hence, introducing weights associated with individual labels in the loss function is a better approach. The loss function with weights is shown as follows

$$L(y, \widehat{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_j \left[-y_{ij} \log \widehat{y}_{ij} - (1 - y_{ij}) \log(1 - \widehat{y}_{ij}) \right] \tag{7.2}$$

If the training procedure is done without weights ($w_j = 1.0$) in the loss function, then *precision* and *recall* becomes poor for label frequencies smaller than mean frequency. Hence the weights for labels with frequency less than the mean frequency need to be increased. Let s_j be the label frequency of the j^{th} label. The weight w_j is given as follows:

$$w_j = \max \left\{ 1, \min \left\{ \frac{mean\,(s)}{s_j}, 5 \right\} \right\} \tag{7.3}$$

where, $mean\,(s)$ is *Mean Label Frequency*. Here, we limit the weights to be in the range of $[1, 5]$ for practical purposes. In this work, the mean label frequency for the genome data is 1100. Thus the 1D CNN based approach using the weighted loss function improves the performance of classification for protein sequences. The learning using the weighted loss function mentioned in Eq. 7.2 results in good overall precision and recall, but the precision and recall for individual labels can still be quite poor.

Algorithm 5: A heuristic approach to improve the performance of multi-label classification

Input : Two models A and B
Output: A new, improved output model

1 Choose two models, A and B, with the best two subset accuracy. **while** *either of A or B has a better of both precision and recall* **do**
2 Find the labels l_{A_p} with individual precision for model $A < fixedthreshold$
3 Find the labels l_{B_p} with individual precision for model $B < fixedthreshold$
4 Find the labels l_{A_r} with individual recall for model $A < fixedthreshold$
5 Find the labels $l_{B_r}$with individual recall for model $B < fixedthreshold$
6 **if** $l_{A_p} \geq l_{B_p}$ **then**
7 Substitute the weights of labels having the precision of model $A < fixedthreshold$ by the weights of the same labels of model B and generate a new model C.
8 **else**
9 Substitute the weights of labels having the precision of model $B < fixedthreshold$ by the weights of the same labels of model A and generate a new model C.
10 **end**
11 **if** $l_{A_r} \geq l_{B_r}$ **then**
12 Substitute the weights of labels having recall of model $A < fixedthreshold$ by the weights of the same labels of model B and generate a new model D.
13 **else**
14 Substitute the weights of labels having recall of model $B < fixedthreshold$ by the weights of the same labels of model A and generate a new model D.
15 **end**
16 Select a better model based on subset accuracy between C and D. Consider the selected model based on better subset accuracy as model B go to step 2.
17 **end**

Hence, to improve the individual precision and individual recall of labels, we propose a heuristic based approach inspired by the harmonic heuristics rules with the 1D-CNN based architecture. This heuristic approach relies mostly on modifying the weights of the loss function for each instance that influences the learning process. After creating various variations, two models based on subset accuracy are selected. Let these two models be named A and B. The idea is to import the weights from a model performing better than another model. Using these models A and B, the

labels having individual precision and recall less than a specific $fixedvalue$ are found. In our experimentation, we have set the $fixedvalue$ 67%. Now in the next subsequent learning steps, the weights of the better models again replace the weights of the same labels in the second model for both individual precision and recall. Thus this proposed heuristic approach generates another set of two new models named C and D. This process can be continued until the desired subset accuracy, precision, and recall is achieved. The overall steps of this proposed heuristic approach are formalized in algorithm 5. The proposed approach is implemented using the PyTorch and the experimentation details are discussed in the next section.

7.4 Experimentation and Results

In this section, we discuss the experimental results and provide a detailed analysis of these results obtained by exhaustive experiments. We first discuss the results and performance of various variations of the proposed deep learning model. Later we discuss the improvement in the results by using the heuristic approach.

7.4.1 Experimental Results and Discussion

We have performed experiments to verify our approach to solve the multi-label classification problem related to genome sequences. As mentioned for this, the 1D-CNN is used for the functional annotation of protein data. The initial configuration of parameters and architecture-related deep convolution model is mentioned in Table 7.1. In this network, adam optimizer is used for optimization. This architecture provides 81.39 % precision, 68.38 % recall, 99.03% AUC, and 29.88 % SA. The value of SA is very poor in this model, and it is improved further by making various variations in the structure of the convolution neural network. All the variations in the experiments are denoted as a model.

As shown in table 7.2, model 1 improves the precision, recall, and AUC, but the SA value is reduced slightly by removing the spatial pooling layer from model 0. The precision, recall, AUC, and SA are reported as 83.09 %, 81.97 %, 99.52 %, and 29.02

Table 7.1: Network architecture and parameters for model 0

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=1024, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=1200)

Table 7.2: Network architecture and parameters for model 1

conv (size=6, stride=1, depth=64, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=64, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
fully connected 1 (units=5000, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=1200)

%, respectively by model 1.

As shown in table 7.3, in model 2, the batch norm is removed from model 0, and its impact is clearly seen in the results that evaluation metrics are worsening by a large margin. The precision, recall, AUC, and SA values provided by model 2 are 13.52%, 10.75 %, 77.84%, and 0.01 % respectively.

In model 3, the activation function ReLU is used, and the rest of the configuration is the same as model 0. The summary of the network architecture of model 3 is shown in table 7.4. This variation performs better than model 2, but the overall performance of this model is not good as the values of precision, recall, AUC, and SA values are

Table 7.3: Network architecture and parameters for model 2

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=1024, activation=prelu)
dropout (p=0.5)
fully connected 2 (units=1200)

Table 7.4: Network architecture and parameters for model 3

conv (size=6, stride=1, depth=128, padding=VALID, activation=relu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=1024, activation=relu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=1200)

42.07%, 26.24%, 94.27%, and 3.64% respectively. The effect of the activation function LeakyReLU is presented by model 4, and it is found that it improves the precision, recall, AUC, and SA values in comparison to model 3. The precision, recall, AUC, and SA are reported as 78.28 %, 65.92 %, 98.82 %, and 27.23 %, respectively, by model 4. The architecture is mentioned in table 7.5.

The increment in the number of neurons in the fully connected layer to 4096 improves the recall and SA when it is compared to model 0, but it improves all metrics precision, recall, AUC, and SA when it is compared to model 4. This model is represented as model 5 in table 7.6. The precision, recall, AUC, and SA are reported as 75.10 %, 72.01 %, 98.91 %, and 30.67 %, respectively, by model 5.

Model 6 is the variation of model 1, and this is described in table 7.7. In this model, one fully connected layer with 2048 units and the PReLU activation function is added. The batch norm is included in this model after the dropout value of 0.5. This

Table 7.5: Network architecture and parameters for model 4

conv (size=6, stride=1, depth=128, padding=VALID, activation=leaky_relu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=leaky_relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=leaky_relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=leaky_relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=leaky_relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=leaky_relu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=1024, activation=leaky_relu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=1200)

Table 7.6: Network architecture and parameters for model 5

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=4096, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=2048, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 3 (units=1200)

model improves the recall and AUC values. The values of precision and SA provided by model 6 are not satisfactory. The precision, recall, AUC, and SA are reported as 79.97 %, 72.09 %, 98.96 %, and 29.96 %, respectively, by model 6.

Model 7 as described in tabel 7.8 is the variation of model 1 with the number of output channels in each layer. This model 7 improves the precision, recall, AUC, and SA values to a significant amount as 87.04%, 86.23%, 99.69%, and 34.57%, respectively. Model 8 represents the 2048 neurons in a fully connected layer with the PReLU activation function of model 7 with the batch norm. This variation declines the performance of model 7 slightly. Model 8 is summarized in table 7.9. The precision, recall, AUC, and SA are reported as 83.47 %, 75.80 %, 99.20 %, and 33.36 %,

Table 7.7: Network architecture and parameters for model 6

conv (size=6, stride=1, depth=64, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=64, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
fully connected 1 (units=5000, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=2048, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 3 (units=1200)

Table 7.8: Network architecture and parameters for model 7

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
fully connected 1 (units=5000, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=1200)

respectively, by model 8.

The change in the dropout value to 0.25 provides a better value of precision, recall, AUC, and SA than in model 8, but the effect of this variation is observed to be slightly poor than model 7 for recall, AUC, and SA. This variation in the convolutional neural network is mentioned as model 9, and its structure is summarized in table 7.10. The precision, recall, AUC, and SA are reported as 87.42 %, 81.50 %, 99.61%, and 33.37%, respectively, by model 9.

Model 10 performs best among all the models from 0 to 10. This model is a variation of model 7 with the value of dropout equal to 0.25. The structure of model 10 is shown in table 7.11. The precision, recall, AUC, and SA are reported as 94.30 %, 92.97 %, 99.87 %, and 44.46%, respectively, by model 10. From table 7.1 to table 7.11

Table 7.9: Network architecture and parameters for model 8

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
fully connected 1 (units=5000, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 2 (units=2048, activation=prelu)
dropout (p=0.5)
batch norm (scale=True)
fully connected 3 (units=1200)

Table 7.10: Network architecture and parameters for model 9

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
spatial pyramid pool (levels=3, divs per level=4)
fully connected 1 (units=4096, activation=prelu)
dropout (p=0.25)
batch norm (scale=True)
fully connected 2 (units=1200)

we have provided the different variations in the proposed network. The experimental results for these variations are summarized in table 7.12 in terms of precision, recall, AUC, and SA. In model 2, the batch norm is removed. Due to this, the performance of the network is significantly degraded. This is due to the occurrence of covariance shift and unable to normalize the output values of each layer. In this way, for the CNN network, it becomes difficult to differentiate the similar type of sequences of proteins.

The performance of model 3 is better in comparison to model 2 because of the dropout layer, but its performance is not significantly improved. The activation function used in model 3 is ReLU which suffers from the dying ReLU problem [168]. The results reported by activation functions PReLU and leaky ReLU are good and significantly better than ReLU as shown in table 7.12 because these activation functions

Table 7.11: Network architecture and parameters for model 10

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max pool (size=2, stride=2, padding=VALID)
batch norm (scale=False)
fully connected 1 (units=5000, activation=prelu)
dropout (p=0.25)
batch norm (scale=True)
fully connected 2 (units=1200)

Table 7.12: Summary of the results

Model	**Precision**	**Recall**	**AUC**	**SA**
0	81.39%	68.38%	99.03%	29.88%
1	83.09%	81.97%	99.52%	29.02%
2	13.52%	10.75%	77.84%	0.01%
3	42.07%	26.24%	94.27%	3.64%
4	78.28%	65.92%	98.82%	27.23%
5	75.10%	72.01%	98.91%	30.67%
6	79.97%	72.09%	98.96%	29.96%
7	87.04%	86.23%	99.69%	34.57%
8	83.47%	75.80%	99.20%	33.36%
9	87.42%	81.50%	99.61%	33.37%
10	94.30%	92.97%	99.87%	44.46%

handle the dying ReLU problem.

Spatial pyramid pooling is used to convert any size input to a fixed size output [118]. It is used between the convolution layer and the fully connected layer. In our experiments, it is used in models 2, 3, 4, 5, and 9 in our experimentation. The performance is improved by removing the spatial pyramid pooling in model 10, as shown in tables 7.11 and 7.12.

Another factor in the performance of the classification is the distribution of frequency of the labels. The UniProt dataset was found skewed when the frequency of the labels was calculated for the entire data set. Label Frequency Distribution is shown in Figure 7.3.

The distribution of Precision and Recall vs Label Frequency is represented in Figure

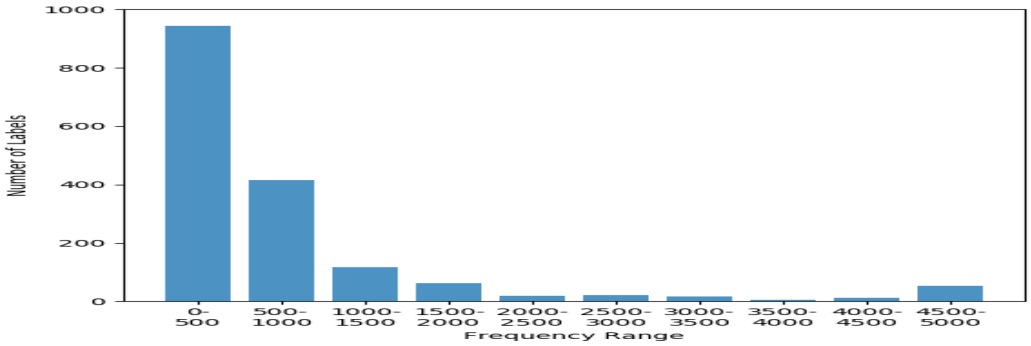


Figure 7.3: Label Frequency Distribution

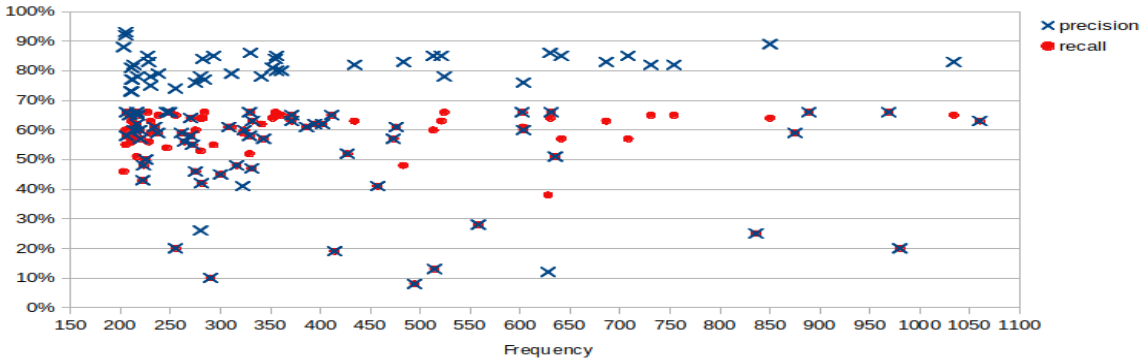


Figure 7.4: Precision and Recall vs Label Frequency

7.4. The heuristic approach is used to improve the performance of the skewed data after the variations using various models. The proposed heuristic approach imports the weights of a better-performing model to another model which have less precision and recall values. Model 10 achieves the subset accuracy of 44.46 % with 225 labels with precision lesser than 67% and 58 labels with recall lesser than 67%(a benchmark set for reference purposes). Another model (Model 11) is trained with no weights in the loss function and achieved subset accuracy of 50.85%, with 94 labels with precision lesser than 67%, and 113 labels with recall lesser than 67%. There is no change done in the architecture during this process. The new models can be generated from the existing models, as mentioned in algorithm 1. We denote the generation of new models as phases. The phases are described below.

(i) In Phase-I of crossing the weights, the individual recall can be improved in Model 11 and individual precision in Model 10. Hence, the weights of Model 10 for the

Table 7.13: Results provided by Model 12

Measure	**value**
Subset Accuracy	48.20 %
number of labels with precision lesser than 67%	106
Number of labels with recall lesser than 67%	107

Table 7.14: Results provided by Model 13

Measure	**value**
Subset Accuracy	49.44 %
number of labels with precision lesser than 67%	173
Number of labels with recall lesser than 67%	58

Table 7.15: Results provided by Model 14

Measure	**value**
Subset Accuracy	50.16%
number of labels with precision lesser than 67%	112
Number of labels with recall lesser than 67%	64

Table 7.16: Results provided by Model 15

Measure	**value**
Subset Accuracy	49.31%
number of labels with precision lesser than 67%	137
Number of labels with recall lesser than 67%	85

Table 7.17: Results provided by Model 16

Measure	**value**
Subset Accuracy	55.43%
number of labels with precision lesser than 67%	81
Number of labels with recall lesser than 67%	87

corresponding labels(labels whose recall for Model 11 is lesser than 67%) are assigned to the same labels in Model 11, represented in Model 12. Similarly, weights of Model 11 are assigned to the same labels in Model 10 for labels whose precision for Model 10 is lesser than 67%, represented in Model 13. The results provided by model 12 and 13 are shown in table 7.13 and table 7.14.

(ii) In Phase-II, the same process is performed between Model 11 and Model 13(because of higher subset accuracy), resulting in Model 14(improving recall in Model 11) and Model 15(improving precision in Model 13). The results provided by model 12 and 13 are shown in table 7.15 and table 7.16.

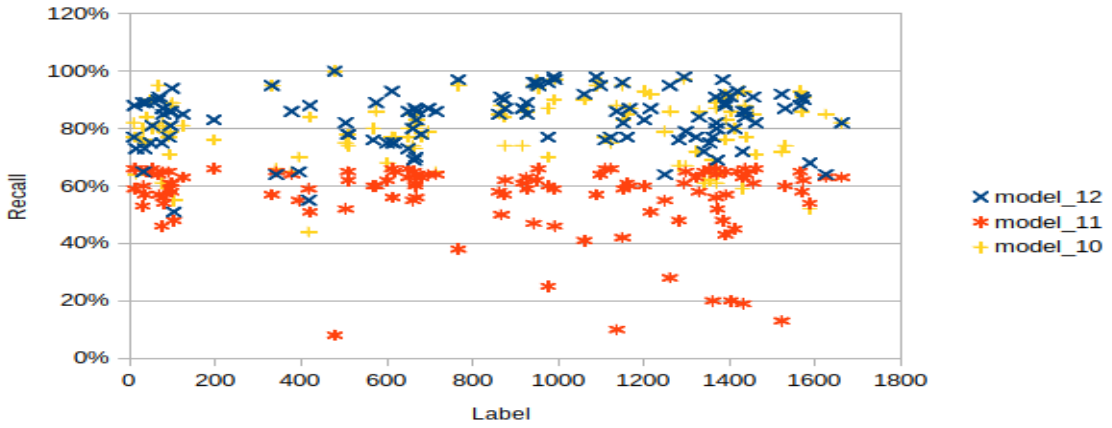


Figure 7.5: Plot of Recall for individual Labels for Models - 10,11 and 12.

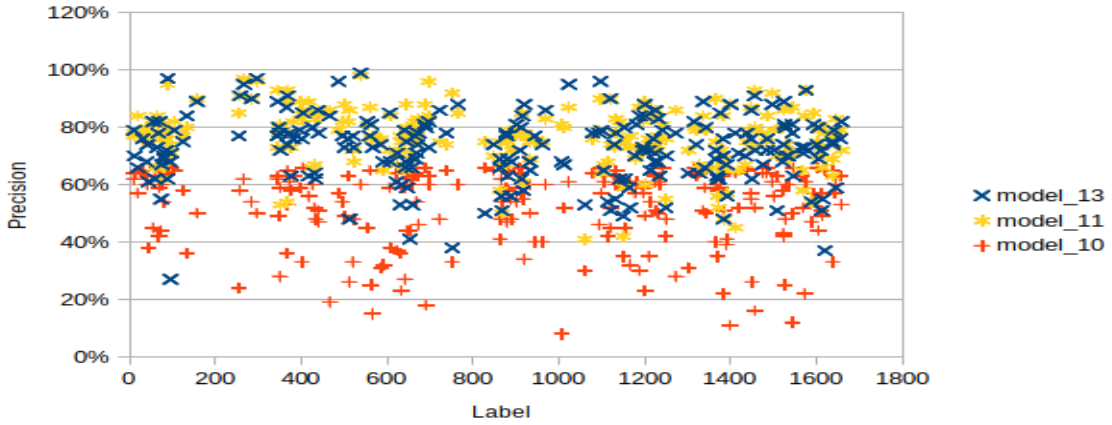


Figure 7.6: Plot of Precision for individual Labels for Models - 10,11 and 13.

(iii) In the final phase, model 11 and model 14 are considered. The results provided by final model 16 are shown in table 7.17.

Figure 7.5 represents the performance of recall of individual label samples versus labels for models 10, 11, and 12. Using the heuristics, model 12 provides better recall. This model 12 is generated by substituting the weights of model 10 in model 11 as per the proposed heuristic approach.

Figure 7.6 provides the visualization of the precision of individual label samples versus label plots for models 10, 11, and 13. Using the heuristics, model 13 provides better precision. This model 13 is generated by substituting the weights of model 11 in model 10 as per the proposed heuristic approach.

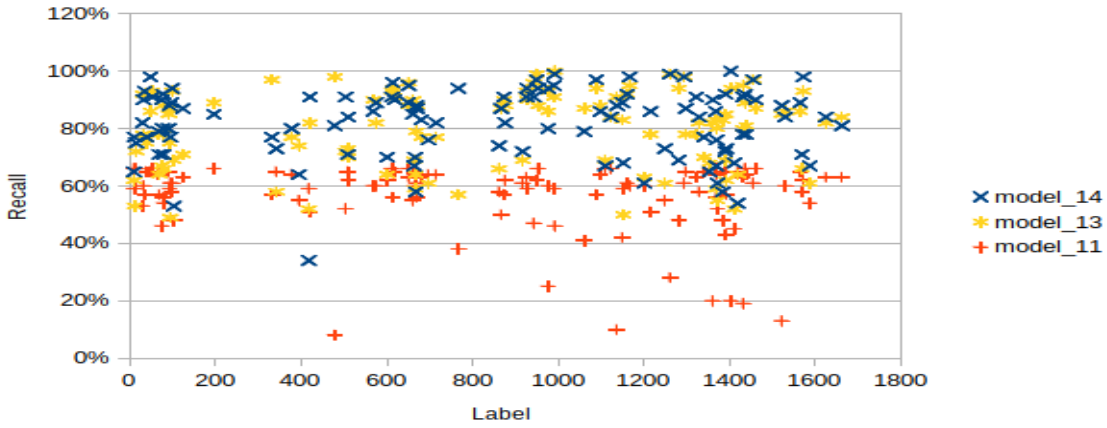


Figure 7.7: Plot of Recall for individual Labels for Models - 11,13 and 14.

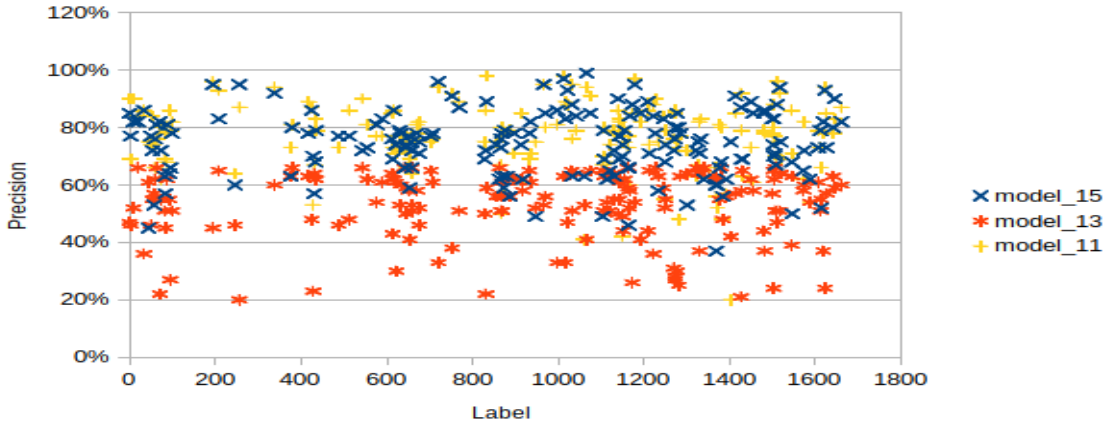


Figure 7.8: Plot of Precision for individual Labels for Models - 11,13 and 15.

Figure 7.7 provides the visualization of recall of individual label samples versus labels for models 11,13,14. Using the heuristics, model 14 provides a better recall. This model 14 is generated by substituting the weights of model 13 in model 11 as per the proposed heuristic approach.

Figure 7.8 provides the visualization of the precision of individual label samples versus labels for model 11,13,15. Using the heuristics, model 15 provides better precision. This model 15 is generated by substituting the weights of model 11 in model 13 as per the proposed heuristic approach.

Figure 7.9 provides the visualization of the precision of individual label samples versus labels for models 11,14,16. Using the heuristics, model 16 provides better

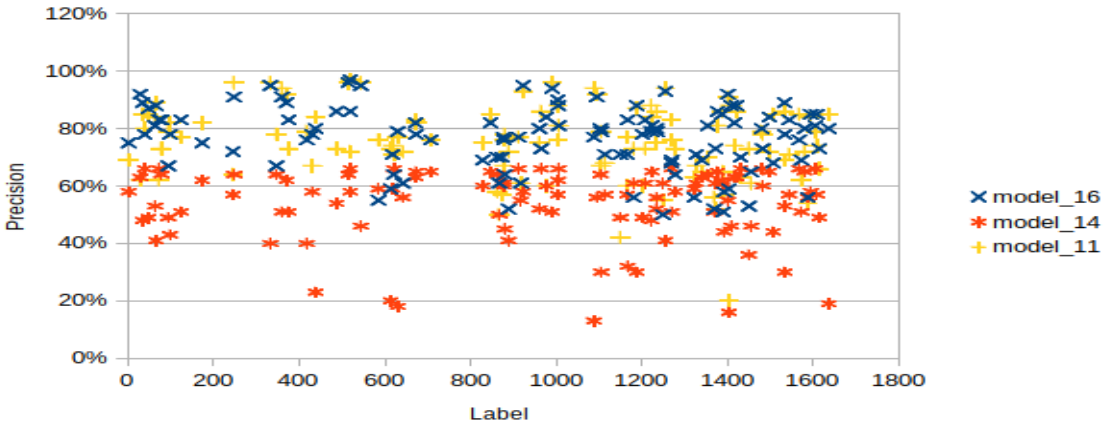


Figure 7.9: Plot of Precision for individual Labels for Models - 11,14 and 16.

Table 7.18: Summary of the results using heuristic approach

Model	Precision	Recall	AUC	SA
11	96.73%	95.23%	99.92%	50.86%
12	96.52%	93.80%	99.92%	48.20%
13	95.46%	95.79%	99.93%	49.44%
14	96.00%	95.56%	99.93%	50.17%
15	95.72%	94.68%	99.92%	49.31%
16	98.37%	97.57%	99.94%	55.43%

precision. This model 16 is generated by substituting the weights of model 11 in model 14 as per the proposed heuristic approach.

Figure 7.5, 7.6, 7.7, 7.8, and 7.9 represent the performance visualization of the heuristic approach by the plots of Precision and Recall of individual label samples for the parent models and the child model. As it can be concluded from the above graphs, performing the substitution of weights using a heuristic approach in the models leads to improvement in Precision and Recall for the individual label samples. All the results produced by models 11 to 16 have been shown in Table 7.18.

The performance of our work has been compared with LSTM [169], biLSTM[169], GRU[169], CNN[169], and SVM[169], and deep CNN[39]. All the compared approaches work on the primary raw structure of protein sequences. The detailed parameter values related to compared approaches are mentioned in table 7.19. These methods report the values of the F1 score for consideration, and the best values are mentioned as a result in Table 7.20. The work based on deep CNN [39] provides performance based on

Table 7.19: Parameter setting for the comparative approaches

Architecture	hidden unit	learning rate	l2 regulizer	dropout
SVM	–	–	–	–
LSTM	100	0.01	0	.85
biLSTM	100	0.007	0	.7
GRU	100	0.01	0	.8
CNN	384	.0001	.0001	.8

DeepCNN	Number of Epochs	10, 20, 30
	Amino Acid Embedding Size	11, 21, 31
	Dropout FC	0.55, 0.65, 0.75, 0.85
	Batch size	64, 128, 256
	Fully Connected Layer Size	256, 512, 1024
	Optimizer	Adam

Table 7.20: comparison of prediction performance across various models

Approaches	Dropout	F1 Score
LSTM	0.85	92.51
biLSTM	0.7	92.22
GRU	0.8	94.84
CNN	0.8	93.4
SVM	NA	87.88
Deep CNN	0.6	97.77
Heuristic based 1D-CNN	not used for heuristic	97.96

the F1 score and AUC. The best AUC for our proposed heuristic approach is 99.94%, which is better than all other competitive approaches. The subset accuracy measure is essential, and we consider it for our results. It can be seen that heuristic-based classifier performs better than other compared approaches. The proposed heuristic-based approach can be investigated with other machine learning or deep learning based approaches in other domains such as computer vision and natural language processing.

7.4.2 Computational Complexity

1D convolution is considered as a sum of the row-wise dot products of input embedding matrix $A \in \mathcal{R}^{W \times L}$ with a filter $K \in \mathcal{R}^{W \times k}$, where W are the dimensionality of word embedding space and k is the length of filter to convolve over the matrix A. One dot product consists of the k multiplication and $k-1$ additions. Total W number of dot products are performed, so the complexity of complete dot product operations

becomes O(Wk). The filter is applied over the input $L - k + 1$ times, where L is the length of the sequence. Usually, $L>>>k$, so the final complexity of a convolution layer becomes O(LWk). The number of the filter in a layer can be denoted by f, so using f filters, the complexity becomes O($LWkf$). The time required by the pooling operation is linear, so the dominating time complexity for each operation is O($LWkf$). The deep neural network requires high computational power due to the size of the data, and this complexity is compensated by the computing resources such as the graphics processing unit(GPU).

7.5 Summary

In this chapter, we have proposed a heuristic based deep learning approach that is able to classify the primary protein sequences along with the handling of data imbalance issue in it. The heuristic approach is used to handle to data imbalance problem of the SwissProt subset of UniProt data. The proposed approach is extensively experimented, and its performance is evaluated in terms of various parameters; and performance metrics precision, recall, subset accuracy, and AUC. The proposed heuristic-based approach handles the data imbalance through weight modulation in the loss function to influence the learning process. The experimental results confirm the effectiveness of the proposed approach by providing the precision, recall, AUC, and subset accuracy as 98.37%, 97.57%, 99.94%, and 55.43%, respectively, which are the best among compared state-of-the-art deep learning approaches.

Chapter 8

Conclusions and Future Work

In this chapter, we have discussed the conclusion and future work related to the proposed multi-label classification approaches throughout in thesis. This thesis primarily investigates the non-iterative and deep learning approaches for multi-label classification. We have developed non-iterative algorithms ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL for multi-label classification. Further, we have developed deep learning based multi-label classifiers in the image and sequential data domain. The developed non-iterative learning algorithm ML-BLS creates the mapped features using the input instances and then creates the enhancement nodes using the mapped features. At the output layer of ML-BLS, the adaptive threshold function is used to separate the relevant label from the irrelevant labels for an input instance. Further, we have developed ML-FBLS which is a hybrid neuro-fuzzy approach for multi-label classification. In the ML-FBLS, the mapped features are replaced by the fuzzy subsystems to get a better interpretation of the results in comparison to ML-BLS. The ML-FBLS improves the performance of ML-BLS in terms of five evaluation metrics of multi-label classification. To improve the evaluation metrics coverage and average precision further, we have proposed a simple and effective SLFN-based ML-RVFL classifier, which connects the input features directly to the output layer using direct links. This approach improves the performance of multi-label classification in comparison to both ML-BLS and ML-FBLS. Finally, we have proposed a fourth non-iterative multi-label classifier ML-KRVFL, which uses the kernel approach to avoid the estimation of the number

of neurons in the hidden layer of ML-RVFL. All of these four multi-label classifiers ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL, use pseudoinverse to compute the output layer weights. Further, for complex domains such as images, these approaches lead to huge time complexity due to the increased overheads in pseudoinverse computation during training; hence we have proposed deep learning based multi-label classifiers for multi-label classification in the image and sequential data domains because deep learning approaches are able to extract features from a large amount of raw data. For the image domain, we have proposed $MLGCN_{pairnorm}$ which uses semantic embeddings of the labels as the nodes of GCN. $MLGCN_{pairnorm}$ is able to tackle the over-smoothing problem of GCN using the simple normalization approach pairnorm. Further, we have extended $MLGCN_{pairnorm}$ to consider the unseen labels by proposing a zero-shot multi-label classification approach $ML-ZSLPGCN$. For the sequential data domain, a large amount of raw sequential data is available to be processed using deep learning. We have proposed a heuristic-based deep learning multi-label classifier to classify the primary protein sequences. This heuristic-based classifier also handles the data imbalance problem which occurs in the primary protein sequence obtained from Uniprot data. In this thesis, all the proposed algorithms are generalized for multi-label classification in their respective domains. The results and analysis exhibit that the proposed approaches outperformed the existing state-of-the-art multi-label algorithms on the basis of evaluation measures and statistical tests in their respective domains

8.1 Summary of Research Achievements

The objectives specified in Section 2.3.2 have been successfully fulfilled by the following main contributions:

(i) **ML-BLS and ML-FBLS multi-label classification approaches:** We have proposed non-iterative randomization-based neural networks named ML-BLS and ML-FBLS for multi-label classification. The output weights of these neural networks are computed using pseudoinverse. At the output layer, multi-label

classification is performed by using an adaptive threshold function. The computation of output weights using pseudoinverse retains the faster computation power of these algorithms compared to iterative learning algorithms. The adaptive threshold function used in the proposed approach considers the correlation among the output labels and the whole dataset for threshold computation. These approaches are simple to implement and less computationally expensive in comparison to iterative algorithms such as BP-MLL and RankSVM.

(ii) **ML-RVFL and ML-KRVFL multi-label classification approaches:** We have proposed simple and effective non-iterative randomization-based neural networks ML-RVFL and ML-KRVFL for multi-label classification. The ML-RVFL is a single-layer feedforward neural network that improves the performance of multi-label classification based on five evaluation measures hamming loss, ranking loss, one error, coverage, and average precision. At the output layer of ML-RVFL, multi-label classification is performed by using an adaptive threshold function. Further, we have proposed a kernelized approach ML-KRVFL which eliminates the need for estimation of the number of neurons in the hidden layer. Overall, ML-KRVFL provides the best performance among all proposed non-iterative classifiers.

(iii) $\boldsymbol{MLGCN_{pairnorm}}$ **multi-label classifier to tackle the over-smoothing problem in GCN for image domain:** To tackle the complex image domain problems, we have proposed GCN based multi-label classifier $MLGCN_{pairnorm}$ which represents the labels as feature embeddings. This approach tackles the over-smoothing problem of the deep structure of GCN by proposing a simple normalization technique pairnorm. Further, it uses a correlation matrix to consider the dependency among labels. Thus it is able to provide scalability for the number of labels as these labels are used in the form of nodes in GCN. Experimental outcomes show that it outperforms existing state-of-the-art approaches in terms of mean average precision, per class precision, per class recall, per class F1 score, overall precision, overall recall, and overall F1 score on the MS-COCO and VOC 2007 datasets.

(iv) ***ML − ZSLPGCN* zero-shot multi-label classifier for image domain:** We have proposed the $ML-ZSLPGCN$ approach for zero-shot multi-label classification using GCN. The proposed approach takes the benefits of seen labels created from the label awareness module and unseen labels from the semantic embedding. Label aware module and semantic embedding both are used to represent the labels feature for seen and unseen labels, respectively, for multi-label zero-shot learning. The over-smoothing issue in the zero-shot multi-label learning is also tackled by introducing pairnorm. Experimentation testing ensures that the proposed classifier $ML - ZSLPGCN$ outperforms existing zero-shot state-of-the-art approaches in terms of precision, recall, and F1 score on the MS-COCO and NUS-WIDE datasets.

(v) **Heuristic-based deep learning multi-label classifier for sequential protein data:** We have proposed a novel heuristic-based approach with deep CNN to improve the overall performance of multi-label classification to functionally annotate the protein sequences, which is sequential data, and this approach is able to handle the issue of data imbalance of the labels related to these protein sequences. The filtering of data (SwissProt subset of the UniProt Dataset) is discussed to make it ready for applying the neural network. For this, filtering methods are applied before encoding the protein sequence for mathematical realization. In this phase, the protein sequences are considered as a mathematical function that takes the sequence as an input and provides the functional classes. We used the encoding of sequences after the prepossessing to make it compatible with the 1D-Convolution Neural Network to classify genome data. Experimental results show that the proposed heuristic based approach performs better than existing state-of-the-art algorithms.

8.2 Future Research Directions

In recent times, multi-label classification has been explored in various research areas such as image, sequential modeling, and zero-shot learning. Despite the signif-

icant progress in multi-label classification, it has the potential to explore in several interesting future directions.

(i) **Extension of Non-iterative learning based multi-label classification to handle large data:** The non-iterative algorithms ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL provide the simple implementation of neural networks for multi-label classification. These approaches provide good results with simple structure. In the future, these approaches can be extended with fuzzy inference and combined with GCN to represent the labels. ML-BLS, ML-FBLS, ML-RVFL, and ML-KRVFL are limited for the preprocessed data, and with medium size datasets due to the pseudoinverse computation, these approaches can be extended with incremental learning to make them suitable for large datasets [170]. In the future, the approaches have scope to be used with intuitionistic fuzzy systems [171], which provide the weightage to the input samples based on their features.

(ii) **Graph Convolution Network based multi-label classifier:** The GCN based approaches $MLGCN_{pairnorm}$ and $ML-ZSLPGCN$ proposed in the thesis are developed for the image domain, and GCN is used to represent the labels with deep learning models such as ResNet 101. In the future, the proposed approach can be extended with other embeddings as graph nodes such as Google News [87] and FastText semantic embeddings [172]. In this thesis, we consider correlation as the inter-dependence relationship for $MLGCN_{pairnorm}$ and $ML-ZSLPGCN$, which can be extended to use the geometry information and prior knowledge as an interdependence relationship among nodes.

(iii) **Multi-label classifier using a heuristic-based deep learning approach for sequential models:** A multi-label classifier based on a heuristic-based deep learning approach is designed to handle the data imbalance problem. The proposed heuristic approach can be extended further to other deep learning based approaches, such as RNN constructing methodologies [173] for multi-label classification. In the future, this approach can be extended with image domain deep

learning models. The other metaheuristic algorithms, such as evolutionary algorithms, can be used to optimize the structure of deep learning models rather than backpropagation to optimize the parameters.

Bibliography

[1] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008, 2010.

[2] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[3] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[4] Min-Ling Zhang, Zhi-Hua Zhou, and Grigorios Tsoumakas. Learning from multi-label data. In *ECML/PKDD*, volume 9, 2009.

[5] Andrew Kachites McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI 99 workshop on text learning*. Citeseer, 1999.

[6] Robert E Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine learning*, 39(2):135–168, 2000.

[7] Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 737–744, 2003.

[8] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. *Advances in neural information processing systems*, 14:681–687, 2001.

[9] Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *European conference on principles of data mining and knowledge discovery*, pages 42–53. Springer, 2001.

[10] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[11] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 17–26, 2007.

[12] Mei Wang, Xiangdong Zhou, and Tat-Seng Chua. Automatic image annotation via local multi-label classification. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 17–26, 2008.

[13] Fadi A Thabtah, Peter Cowling, and Yonghong Peng. Mmac: A new multi-class, multi-label associative classification approach. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 217–224. IEEE, 2004.

[14] Adriano Veloso, Wagner Meira, Marcos Gonçalves, and Mohammed Zaki. Multi-label lazy associative classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 605–612. Springer, 2007.

[15] Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. Maximal margin labeling for multi-topic text categorization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, pages 649–656, 2004.

[16] Lei Tang, Suju Rajan, and Vijay K Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, pages 211–220, 2009.

[17] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18, page 5. Citeseer, 2008.

[18] Yang Song, Lu Zhang, and C Lee Giles. A sparse gaussian processes classification framework for fast tag suggestions. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 93–102, 2008.

[19] Lingqiao Liu, Peng Wang, Chunhua Shen, Lei Wang, Anton Van Den Hengel, Chao Wang, and Heng Tao Shen. Compositional model based fisher vector coding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2335–2348, 2017.

[20] Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of random forests for large-scale image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(3):490–503, 2015.

[21] Chris Sanden and John Z Zhang. Enhancing multi-label music genre classification through ensemble techniques. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 705–714, 2011.

[22] Nam-phuong Nguyen, Michael Nute, Siavash Mirarab, and Tandy Warnow. HIPPI: highly accurate protein family classification with ensembles of HMMs. *BMC genomics*, 17(10):89–100, 2016.

[23] Natalie Dawson, Ian Sillitoe, Russell L Marsden, and Christine A Orengo. The classification of protein domains. In *Bioinformatics*, pages 137–164. Springer, 2017.

[24] Fangfang Luo, Wenzhong Guo, Yuanlong Yu, and Guolong Chen. A multi-label classification algorithm based on kernel extreme learning machine. *Neurocomputing*, 260:313–320, 2017.

[25] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[26] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[27] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[28] Ponnuthurai Nagaratnam Suganthan. On non-iterative learning algorithms with closed-form solution. *Applied Soft Computing*, 70:1078–1082, 2018.

[29] CL Philip Chen and Zhulin 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(1):10–24, 2017.

[30] Vikas Chauhan and Aruna Tiwari. On the Construction of Hierarchical Broad Learning Neural Network: An Alternative Way of Deep Learning. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 182–188. IEEE, 2018.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] Rajasekar Venkatesan and Meng Joo Er. Multi-label classification method based on extreme learning machines. In *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 619–624. IEEE, 2014.

[34] Yunchao Gong, Yangqing Jia, Alexander Toshev, Thomas Leung, and Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. In *International Conference on Learning Representations*, 2014.

[35] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Repre-*

sentations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[36] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[37] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek F. Abdelzaher. Revisiting "over-smoothing" in deep gcns. *CoRR*, abs/2003.13663, 2020.

[38] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[39] Da Zhang and Mansur R Kabuka. Protein family classification from scratch: A cnn based deep learning approach. *IEEE/ACM transactions on computational biology and bioinformatics*, 18(5):1996–2007, 2020.

[40] Guangjin Ou, Guoxian Yu, Carlotta Domeniconi, Xuequan Lu, and Xiangliang Zhang. Multi-label zero-shot learning with graph convolutional networks. *Neural Networks*, 132:333–341, 2020.

[41] Zong Woo Geem. *Music-inspired harmony search algorithm: theory and applications*, volume 191. Springer, 2009.

[42] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 2018.

[43] Balázs Szalkai and Vince Grolmusz. Near perfect protein multi-label classification with deep neural networks. *Methods*, 132:50–56, 2018.

[44] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[45] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[46] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE transactions on knowledge and data engineering*, 23(7):1079–1089, 2010.

[47] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European conference on machine learning*, pages 406–417. Springer, 2007.

[48] Zhaomin Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Learning Graph Convolutional Networks for Multi-Label Recognition and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[49] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[50] Yohhan Pao. Adaptive pattern recognition and neural networks. 1989. Reading, MA (US); Addison-Wesley Publishing Co., Inc.

[51] Le Zhang and Ponnuthurai N Suganthan. A comprehensive evaluation of random vector functional link networks. *Information sciences*, 367:1094–1105, 2016.

[52] Rakesh Katuwal and Ponnuthurai N Suganthan. Stacked autoencoder based deep random vector functional link neural network for classification. *Applied Soft Computing*, 85:105854, 2019.

[53] Yongshan Zhang, Jia Wu, Zhihua Cai, Bo Du, and S Yu Philip. An unsupervised parameter learning model for RVFL neural network. *Neural Networks*, 112:85–97, 2019.

[54] Najdan Vuković, Milica Petrović, and Zoran Miljković. A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression. *Applied Soft Computing*, 70:1083–1096, 2018.

[55] Ponnuthurai N Suganthan and Rakesh Katuwal. On the origins of randomization-based feedforward neural networks. *Applied Soft Computing*, 105:107239, 2021.

[56] Filippo Maria Bianchi and Ponnuthurai Nagaratnam Suganthan. Non-iterative learning approaches and their applications. *Cognitive Computation*, 12(2):327–329, 2020.

[57] Rajasekar Venkatesan, Meng Joo Er, Shiqian Wu, and Mahardhika Pratama. A novel online real-time classifier for multi-label data streams. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1833–1840. IEEE, 2016.

[58] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, page 515–521, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[59] Li-Xin Wang and Chen Wei. Approximation accuracy of some neuro-fuzzy approaches. *IEEE transactions on fuzzy systems*, 8(4):470–478, 2000.

[60] V Suresh Kumar, S Vijaya Chandra, and C Susil Kumar. Neuro-Fuzzy Function Approximations Using Feedforward Networks-An Application of Sigmoidal Signal. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 895–898. IEEE, 2010.

[61] Yoichi Hayashi and James J Buckley. Approximations between fuzzy expert systems and neural networks. *International Journal of Approximate Reasoning*, 10(1):63–73, 1994.

[62] JJ Buckley and Yoichi Hayashi. Numerical relationships between neural networks, continuous functions, and fuzzy systems. *Fuzzy Sets and Systems*, 60(1):1–8, 1993.

[63] James J Buckley, Yoichi Hayashi, and Ernest Czoga\la. On the equivalence of neural nets and fuzzy expert systems. *Fuzzy Sets and Systems*, 53(2):129–134, 1993.

[64] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[65] Mohua Banerjee, Sushmita Mitra, and Sankar K Pal. Rough fuzzy MLP: Knowledge encoding and classification. *IEEE Transactions on Neural Networks*, 9(6):1203–1216, 1998.

[66] Sushmita Mitra, Rajat K De, and Sankar K Pal. Knowledge-based fuzzy MLP for classification and rule generation. *IEEE Transactions on Neural Networks*, 8(6):1338–1350, 1997.

[67] Li-Min Fu. Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):173–182, 1993.

[68] Yu-Lin He, Cheng-Hao Wei, Hao Long, Rana Aamir Raza Ashfaq, and Joshua Zhexue Huang. Random weight network-based fuzzy nonlinear regression for trapezoidal fuzzy number data. *Applied soft computing*, 70:959–979, 2018.

[69] Shuang Feng and CL Philip Chen. Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification. *IEEE transactions on cybernetics*, 50(2):414–424, 2018.

[70] C Radhakrishna Rao. Generalized inverse of a matrix and its applications. In *Vol. 1 Theory of Statistics*, pages 601–620. University of California Press, 1972.

[71] Denis Serre. Matrices: Theory & applications additional exercises. *L'Ecole Normale Supérieure de Lyon*, 2001.

[72] J Mercer. Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations. *Philos. Trinsdictions Rogyal Soc*, 209:4–415, 1909.

[73] Yining Li, Chen Huang, Chen Change Loy, and Xiaoou Tang. Human attribute recognition by deep hierarchical contexts. In *European Conference on Computer Vision*, pages 684–700. Springer, 2016.

[74] Zongyuan Ge, Dwarikanath Mahapatra, Suman Sedai, Rahil Garnavi, and Rajib Chakravorty. Chest X-rays classification: A multi-label and fine-grained problem. *arXiv preprint arXiv:1807.07247*, 2018.

[75] Marian George and Christian Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. In *European Conference on Computer Vision*, pages 440–455. Springer, 2014.

[76] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[77] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[78] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[80] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[81] Qiang Li, Maoying Qiao, Wei Bian, and Dacheng Tao. Conditional graphical lasso for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2977–2986, 2016.

[82] Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. Multi-label image recognition by recurrently discovering attentional regions. In *Proceedings of the IEEE international conference on computer vision*, pages 464–472, 2017.

[83] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5513–5522, 2017.

[84] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3538–3545, 2018.

[85] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[86] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand . Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[87] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, pages 1–12, 2013.

[88] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *2009 IEEE 12th international conference on computer vision*, pages 309–316. IEEE, 2009.

[89] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90(1):88–105, 2010.

[90] Jason Weston, Samy Bengio, and Nicolas Usunier. WSABIE: scaling up to large vocabulary image annotation. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2764–2770. IJCAI, 2011.

[91] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[92] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[93] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

[94] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1576–1585, 2018.

[95] Xuanwu Liu, Zhao Li, Jun Wang, Guoxian Yu, Carlotta Domenicon, and Xiangliang Zhang. Cross-modal zero-shot hashing. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 449–458. IEEE, 2019.

[96] Yue Zhu, James T Kwok, and Zhi-Hua Zhou. Multi-label learning with global and local label correlation. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1081–1094, 2017.

[97] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 410–419. SIAM, 2008.

[98] Qiaoyu Tan, Yanming Yu, Guoxian Yu, and Jun Wang. Semi-supervised multi-label classification using incomplete label information. *Neurocomputing*, 260:192–202, 2017.

[99] Guoxian Yu, Xia Chen, Carlotta Domeniconi, Jun Wang, Zhao Li, Zili Zhang, and Xindong Wu. Feature-induced partial multi-label learning. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1398–1403. IEEE, 2018.

[100] Eva Gibaja and Sebastián Ventura. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):1–38, 2015.

[101] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438, 2015.

[102] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2927–2936, 2015.

[103] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomás Mikolov. Devise: A deep visual-semantic embedding model. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information*

Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 2121–2129, 2013.

[104] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):453–465, 2013.

[105] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and others. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4247–4255, 2015.

[106] Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2441–2448, 2014.

[107] Yanwei Fu, Yongxin Yang, Tim Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-label zero-shot learning. *arXiv preprint arXiv:1503.07790*, 2015.

[108] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning*, pages 2152–2161. PMLR, 2015.

[109] Abhilash Gaure, Aishwarya Gupta, Vinay Kumar Verma, and Piyush Rai. A probabilistic framework for zero-shot multi-label learning. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 1, page 3, 2017.

[110] P. J. Halling. Proteins: Structures and molecular properties (2nd edition). by thomas e. creighton, w. h. freeman, new york, 1992. *Journal of Chemical Technology & Biotechnology*, 62(1):105–105, 1995.

[111] Nurul Nadzirin and Mohd Firdaus-Raih. Proteins of unknown function in the Protein Data Bank (PDB): an inventory of true uncharacterized proteins and

computational tools for their analysis. *International journal of molecular sciences*, 13(10):12761–12772, 2012.

[112] Temple F Smith, Michael S Waterman, and others. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[113] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[114] Sean R Eddy. Accelerated profile HMM searches. *PLoS computational biology*, 7(10):e1002195, 2011.

[115] Kristoffer Illergård, David H Ardell, and Arne Elofsson. Structure is three to ten times more conserved than sequence—a study of structural response in protein cores. *Proteins: Structure, Function, and Bioinformatics*, 77(3):499–508, 2009.

[116] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[117] Seokjun Seo, Minsik Oh, Youngjune Park, and Sun Kim. DeepFam: deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics*, 34(13):i254–i262, 2018.

[118] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[119] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[120] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[121] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[122] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[123] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[124] Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1504–1512, Cambridge, MA, USA, 2015. MIT Press.

[125] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, volume 21, pages 53–59, 2008.

[126] Pinar Duygulu, Kobus Barnard, Joao FG de Freitas, and David A Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *European conference on computer vision*, pages 97–112. Springer, 2002.

[127] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *2008 eighth IEEE international conference on data mining*, pages 995–1000. IEEE, 2008.

[128] John Pestian, Chris Brew, Pawel Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Wlodzislaw Duch. A shared task involving multi-

label classification of clinical free text. In *Biological, translational, and clinical language processing*, pages 97–104, 2007.

[129] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook, 2nd ed*, pages 667–685. Springer, 2010.

[130] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[131] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009.

[132] UniProt Consortium. The universal protein resource (UniProt) 2009. *Nucleic acids research*, 37(suppl_1):D169–D174, 2009.

[133] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[134] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[135] CL Philip Chen and Zhulin 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)*, pages 1271–1276. IEEE, 2017.

[136] CL Philip Chen, Zhulin Liu, and Shuang Feng. Universal approximation capability of broad learning system and its structural variations. *IEEE transactions on neural networks and learning systems*, 30(4):1191–1204, 2018.

[137] Tong Zhang, Zhu-lin Liu, Xue-Han Wang, Xiao-Fen Xing, CL Philip Chen, and Enhong Chen. Facial expression recognition via broad learning system. In *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 1898–1902. IEEE, 2018.

[138] Ebrahim H Mamdani. Advances in the linguistic synthesis of fuzzy controllers. *International Journal of Man-Machine Studies*, 8(6):669–678, 1976.

[139] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, SMC-15(1):116–132, 1985.

[140] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

[141] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier]. *IEEE Computational Intelligence Magazine*, 12(4):61–72, 2017.

[142] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[143] Xi-Zhu Wu and Zhi-Hua Zhou. A unified view of multi-label performance measures. In *International Conference on Machine Learning*, pages 3780–3788. PMLR, 2017.

[144] Yuncheng Li, Yale Song, and Jiebo Luo. Improving pairwise ranking for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3617–3625, 2017.

[145] Mei-Chen Yeh and Yi-Nan Li. Multilabel deep visual-semantic embedding. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1530–1536, 2019.

[146] Yunchao Wei, Wei Xia, Min Lin, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan. HCP: A flexible CNN framework for multi-label image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1901–1907, 2015.

[147] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[148] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019.

[149] Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Order-free rnn with visual attention for multi-label classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[150] Weifeng Ge, Sibei Yang, and Yizhou Yu. Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1277–1286, 2018.

[151] Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, and Jianfeng Lu. Multilabel image classification with regional latent semantic dependencies. *IEEE Transactions on Multimedia*, 20(10):2801–2813, 2018.

[152] Hao Yang, Joey Tianyi Zhou, Yu Zhang, Bin-Bin Gao, Jianxin Wu, and Jianfei Cai. Exploit bounding box annotations for multi-label object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–288, 2016.

[153] Tianshui Chen, Zhouxia Wang, Guanbin Li, and Liang Lin. Recurrent attentional reinforcement learning for multi-label image recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[154] Guo-Sen Xie, Li Liu, Xiaobo Jin, Fan Zhu, Zheng Zhang, Jie Qin, Yazhou Yao, and Ling Shao. Attentive region embedding network for zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9384–9393, 2019.

[155] Jie Song, Chengchao Shen, Yezhou Yang, Yang Liu, and Mingli Song. Transductive unbiased embedding for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1024–1033, 2018.

[156] Yang Zhang, Boqing Gong, and Mubarak Shah. Fast zero-shot image tagging. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5985–5994. IEEE, 2016.

[157] Raymond E Wright. Logistic regression. *Reading and Understanding Multivariate Statistics*, pages 217–244, 1995.

[158] Hang Shao, Yuchen Guo, Guiguang Ding, and Jungong Han. Zero-shot multi-label learning via label factorisation. *IET Computer Vision*, 13(2):117–124, 2019.

[159] Edgardo A Ferran, Pascual Ferrara, and Bernard Pflugfelder. Protein classification using neural networks. In *ISMB*, pages 127–135, 1993.

[160] Cathy H Wu. Neural networks for molecular sequence classification. In *The Protein Folding Problem and Tertiary Structure Prediction*, pages 279–305. Springer, 1994.

[161] Xiujuan Lei, Xiaoqin Yang, and Hamido Fujita. Random walk based method to identify essential proteins by integrating network topology and biological characteristics. *Knowledge-Based Systems*, 167:53–67, 2019.

[162] Xiujuan Lei, Yulian Ding, Hamido Fujita, and Aidong Zhang. Identification of dynamic protein complexes based on fruit fly optimization algorithm. *Knowledge-Based Systems*, 105:270–277, 2016.

[163] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[164] Xueliang Liu. Deep recurrent neural network for protein function prediction from sequence. *arXiv preprint arXiv:1701.08318*, 2017.

[165] Swagatam Das, Arpan Mukhopadhyay, Anwit Roy, Ajith Abraham, and Bijaya K Panigrahi. Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):89–106, 2010.

[166] Ren Diao and Qiang Shen. Feature selection with harmony search. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(6):1509–1523, 2012.

[167] Duc Chinh Hoang, Parikshit Yadav, Rajesh Kumar, and Sanjib Kumar Panda. Real-time implementation of a harmony search algorithm-based clustering protocol for energy-efficient wireless sensor networks. *IEEE transactions on industrial informatics*, 10(1):774–783, 2013.

[168] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[169] Timothy K Lee and Tuan Nguyen. Protein family classification with neural networks. *Accessed: Dec*, 10:2018, 2016.

[170] Vikas Chauhan and Aruna Tiwari. Randomized neural networks for multilabel classification. *Applied Soft Computing*, 115:108184, 2022.

[171] Ashwani Kumar Malik, MA Ganaie, M Tanveer, PN Suganthan, Alzheimer's Disease Neuroimaging Initiative Initiative, et al. Alzheimer's disease diagnosis via intuitionistic fuzzy random vector functional link network. *IEEE Transactions on Computational Social Systems*, 2022.

[172] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomás Mikolov. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651, 2016.

[173] Ziming Zhang, Guojun Wu, Yanhua Li, Yun Yue, and Xun Zhou. Deep incremental rnn for learning sequential data: A lyapunov stable dynamical system. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 966–975. IEEE, 2021.