Anomaly-Resilient and Robust Learning Frameworks for Real-time QoS Prediction

A THESIS

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

by

Suraj Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
February 2026

INDIAN INSTITUTE OF TECHNOLOGY INDORE

I hereby certify that the work which is being presented in the thesis entitled **Anomaly-Resilient and Robust Learning Frameworks for Real-time QoS Prediction** 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 2023 to February 2026 under the supervision of Dr. Soumi Chattopadhyay, Assistant Professor, Computer Science and Engineering.

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.

Suraj 16.02.26

Signature of the student with date

Suraj Kumar

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

Soumi 16/2/26

Signature of Thesis Supervisor with date

Dr. Soumi Chattopadhyay

Suraj Kumar has successfully given his/her Ph.D. Oral Examination held on **<Date of PhD Oral Examination>**. 28/05/2026

Soumi 29/05/2026

Signature of Thesis Supervisor with date

Dr. Soumi Chattopadhyay

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Soumi Chattopadhyay, for her patient guidance, continuous support, and invaluable mentorship throughout my PhD journey. Her insights, encouragement, and unwavering belief in my abilities helped me build strong research fundamentals and shaped both my academic and personal growth.

I sincerely thank Dr. Chandranath Adak, Assistant Professor at IIT Patna, for his collaboration, support, and providing me the opportunity to work as a visiting scholar at IIT Patna. His guidance in multiple projects enriched my research experience and inspired me through his professionalism, politeness, and dedication to education.

I am deeply grateful to Dr. Arani Bhattacharya, Assistant Professor at IIIT Delhi, for the opportunity to undertake a research internship under his supervision. His research mindset and insightful suggestions greatly influenced my research perspective. I also thank his lab students—Shubham, Vijay, Sapna, Nazia, Jyoti, and Sumit—for their support and memorable moments during my time at IIIT Delhi.

I sincerely thank my Doctoral PSPC Committee members, Prof. Vimal Bhatia and Dr. Nagendra Kumar, for their continuous evaluation, constructive feedback, and encouragement, which helped strengthen the quality and direction of my research.

I would like to thank Prof. Aruna Tiwari for her guidance and for teaching Machine Learning, which laid a strong foundation for my research. I am grateful to Dr. Ranveer Singh for being an exceptional teacher and mentor. His lectures on Linear Algebra strengthened my mathematical foundation, and his leadership as Head of the Department inspired a positive academic environment. I also thank Dr. Puneet Gupta for teaching Pattern Recognition and for his guidance. I sincerely acknowledge his students, Dr. Anup K. Gupta and Trishna Saikia, for their help and encouragement.

I gratefully acknowledge Dr. Aditya Nigam, IIT Mandi, my M.Tech co-supervisor and a major source of inspiration for pursuing a Ph.D. His mentorship and encouragement played a crucial role in shaping my research journey.

I sincerely thank Dr. Moumita Roy, IIIT Guwahati, for her support and encour-

agement during the initial phase of my research journey. I also thank Arpit S. Yadav, Dr. Mehbub Alam, Dr. Indrajit Kalita, Dr. Anand, and the Gymkhana Council for their support and wonderful memories and for creating a welcoming and supportive community at IIIT Guwahati.

I extend my sincere gratitude to my collaborators and colleagues from IIT Patna, especially Utsav K. Nareti, for his constant support, insightful discussions, and assistance during my stay. I also thank Priya, Aritra, and Saba for their research collaboration, and Bibek for the memorable and joyful moments.

I would like to thank my lab colleagues and juniors, Soumya Pandey, Khusi Khosta, Rupa Shree, Anand Suralkar and Suraj Ahirwar, for their support. Special thanks to Arvind Kumar, Ashutosh Parihar and Monish Raj for their help, support and for the meaningful conversations we shared over meals and walks.

I sincerely thank Kuldeep Pathak and Rinku Sir for the insightful conversations on linear algebra, Hindi poetry, and the badminton sessions during the early phase of my PhD. Many thanks to Saurabh Saini and Prashant Sahu for their support and encouragement. I also thank Haresh K. Jadhav, Saurav Sharma (BSBE), and Nisheet Sir for their help and support. I am especially grateful to Ankush Agarwal, my friend since school, for his friendship, support, and many meaningful conversations.

My heartfelt gratitude goes to my family. I am deeply indebted to my father for his unconditional love, sacrifices, and constant encouragement. I thank my sisters, Sapna and Rama, for their unwavering support and belief in me. I also thank my brother-in-law, Mr. Nishant Chaudhary, and my nephew Purvansh, whose presence brought immense joy and motivation.

Last but not least, I express my heartfelt gratitude to my dear friend Rimpi Borah for standing by me during difficult times and supporting me throughout this journey. Her encouragement and belief in me gave me strength when it was needed most.

This journey would not have been possible without the support of all these wonderful individuals. I sincerely thank everyone who contributed, directly or indirectly, to my PhD journey.

Suraj Kumar

Abstract

Quality of Service (QoS) prediction plays a crucial role in ensuring reliable and efficient operation of modern distributed digital services. Accurate estimation of performance attributes, such as response time, throughput, reliability, and availability, is essential for maintaining service-level guarantees, optimizing resource allocation, and enhancing user experience. These attributes directly influence system performance and user satisfaction. In many applications, including cloud platforms, edge computing, and intelligent transportation systems, accurate QoS estimation is critical for ensuring stable, efficient, and safe system operations.

Despite significant research progress, accurate QoS prediction remains challenging due to the inherent characteristics of real-world service interaction data, particularly extreme sparsity. In practice, the user–service interaction matrix is highly incomplete, as users typically invoke only a small subset of available services. As a result, observed QoS entries represent only a limited portion of all possible interactions, restricting reliable supervision and weakening correlation learning. This sparsity significantly limits model generalization to unseen or infrequently invoked services, complicating the development of robust and reliable QoS prediction frameworks.

First, a time-aware QoS prediction framework, termed "Temporal QoS Prediction using Multi-Source Collaborative Features (TPMCF)", is proposed to model dynamic QoS behavior by integrating graph convolutional matrix factorization for collaborative spatial feature extraction with a predictive transformer encoder for temporal dependency learning. This framework effectively captures both structural relationships among user–service interactions and their temporal evolution, while incorporating outlier-aware learning mechanisms to improve prediction robustness.

To address reliability challenges in real-time QoS prediction, an "Anomaly Resilient Real-time QoS Prediction Framework with Graph Convolution (ARRQP)" is proposed. ARRQP employs a multi-head graph convolutional matrix factorization architecture with residual learning to enhance feature representation under sparse interaction conditions. Dedicated modules for grey-sheep, outlier, and cold-start handling

explicitly address anomalous observations, improving prediction stability in static QoS environments.

Building upon these advances, this thesis proposes the "Anomaly Resilient Temporal QoS Prediction using Hypergraph Convoluted Transformer Network (HCTN)", a unified architecture for anomaly-resilient temporal QoS prediction. Unlike conventional graph-based models, HCTN leverages hypergraph convolution to capture higher-order dependencies among users, services, and temporal contexts, enabling more expressive structural representations. A transformer-based temporal module and credibility-aware learning further improve robustness under anomalous and dynamic service conditions.

Finally, to address the limitations of single-metric QoS modeling, this thesis introduces "Sparsely-gated Hierarchical Adaptive Routing for Joint Prediction of QoS (SHARP-QoS)", a joint QoS prediction framework based on sparsely-gated hierarchical routing and hyperbolic graph convolution. By modeling hierarchical dependencies in hyperbolic space and enabling adaptive feature sharing across tasks, SHARP-QoS facilitates stable and efficient multi-QoS prediction while reducing negative transfer. Additionally, an exponential moving average-based loss balancing strategy improves convergence stability and overall generalization.

Extensive experimental evaluation on multiple benchmark datasets, including WS-DREAM and a real-world in-house gRPC autonomous vehicle (AV) service dataset, demonstrates that the proposed frameworks consistently outperform state-of-the-art methods in prediction accuracy, anomaly robustness, scalability, and computational efficiency. Ablation and statistical analyses further validate the effectiveness of the proposed architectural components and learning strategies.

Overall, this thesis establishes a comprehensive foundation for anomaly-resilient QoS prediction by integrating graph learning, hypergraph modeling, transformer-based temporal modeling, and hierarchical multi-task optimization. The proposed frameworks significantly enhance prediction reliability and adaptability in dynamic service environments, enabling practical and dependable service selection and management in modern distributed systems.

List of Publications

Publications from the PhD Thesis Work

Published

1. **S. Kumar**, S. Chattopadhyay and C. Adak, "Anomaly-Resilient Temporal QoS Prediction Using Hypergraph Convoluted Transformer Network," in **IEEE Transactions on Network and Service Management**, vol. 23, pp. 3556-3568, 2026, doi: 10.1109/TNSM.2026.3674650.

2. **S. Kumar** and S. Chattopadhyay, "ARRQP: Anomaly Resilient Real-Time QoS Prediction Framework With Graph Convolution," in **IEEE Transactions on Services Computing**, vol. 18, no. 3, pp. 1245-1261, May-June 2025, doi: 10.1109/TSC.2025.3565376.

3. **S. Kumar**, S. Chattopadhyay and C. Adak, "TPMCF: Temporal QoS Prediction Using Multi-Source Collaborative Features," in **IEEE Transactions on Network and Service Management**, vol. 21, no. 4, pp. 3945-3955, Aug. 2024, doi: 10.1109/TNSM.2024.3395428.

Communicated Articles

1. **S. Kumar**, A. Kumar, S. Chattopadhyay, SHARP-QoS: Sparsely-gated Hierarchical Adaptive Routing for Joint Prediction of QoS, under review with IEEE Transactions on Services Computing. `https://arxiv.org/pdf/2512.17262`.

Other outcomes outside of the Ph.D. thesis

Published

1. U. K. Nareti, S. Chattopadhyay, P. Mallick, C. Adak, **S. Kumar**, A. V. Daga, A. Wase, and A. Roy, An Adaptive Data-Resilient Multi-Modal Framework for Hierarchical Multi-Label Book Genre Identification, accepted with IEEE Transactions on Multimedia. `https://arxiv.org/pdf/2505.03839`

2. K. Priya, **S. Kumar**, A. Dey, C. Adak, S. Chattopadhyay, S. Chanda and S. Marinai, Graph Convolutional Teacher-Student Framework for Writer Inspection from Handwritten Words with Intra-Writer Variability, in 19th IAPR **International Conference on Document Analysis and Recognition (ICDAR)**, Springer, 2025.

Communicated Articles

1. **S. Kumar**, M. Raj, U. K. Nareti, S. Akram, S. Chattopadhyay, C. Adak, M. Saqib, Adaptive Modality Reasoning for Efficient Lightweight Multimodal Intent Recognition, under review with double blind policy.

2. **S. Kumar**, U. K. Nareti, S. Chattopadhyay, C. Adak, and P. Mallick, Blurb-Refined Inference from Crowdsourced Book Reviews using Hierarchical Genre Mining with Dual-Path Graph Convolutions, under review with double blind policy. `https://arxiv.org/pdf/2512.21076`.

3. U. K. Nareti, **S. Kumar**, S. Pandey, S. Chattopadhyay, and C. Adak, ProtoSi-Tex: Learning Semi-Interpretable Prototypes for Multi-label Text Classification, under review with IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI). `https://arxiv.org/pdf/2510.12534`.

4. R. Bukke, S. Pandey, **S. Kumar**, S. Chattopadhyay, and C. Adak, Agentic Multi-Persona Framework for Evidence-Aware Fake News Detection, under review with IEEE Transactions on Computational Social Systems. `https://arxiv.org/pdf/2512.21039`.

Table of Contents

List of Figures

List of Tables

List of Symbols and Notations

n, m, T	Number of users, services, and discrete time steps
$\mathcal{U}, \mathcal{S}$	Set of users and set of services
$\mathcal{Q}$	QoS interaction matrix / tensor
$q, \hat{q}$	Observed and Predicted QoS value
u_i, s_j	Target user–service pairs
$\mathcal{C}^u, \mathcal{C}^s$	Contextual information of users and services
v	A vector
$\mathbb{Z}, \mathbb{R}$	Set of integers and real numbers
W, b, σ_a	Weight, bias, and activation function
$\mathcal{F}_u, \mathcal{F}_s$	User and service feature matrix
$\mathcal{F}_{St}, \mathcal{F}_Q$	Statistical and QoS-based features
$\mathcal{F}_{Si}, \mathcal{F}_C$	Similarity and Contextual features
$N_l, \mathcal{F}_l$	Number of layers and Feature at layer l
d_q, d_s	Feature dimension of QoS-based and similarity-based features
d_c	Feature dimension of contextual features
$\mathcal{X}_u, \mathcal{X}_f$	Low-rank user and service features via matrix factorization

$\mathcal{G}, \mathbb{G}$	QoS Graph and Hypergraph
$\mathcal{G}^r, \mathbb{G}^r$	Region Graph and Hypergraph
$\mathcal{G}^a, \mathbb{G}^a$	Autonomous systems Graph and Hypergraph
V, E	Set of Vertices and Edges in G
$\mathbb{V}, \mathbb{E}$	Set of Vertices and Edges in G
$\mathcal{A}, \bar{\mathcal{A}}$	Adjacency and Normalized adjacency matrix, respectively
$\bar{\mathbb{A}}$	Normalized hypergraph adjacency matrix
$\mathcal{E}$	Node embedding
$\mathcal{D}, \mathbb{I}$	Diagonal degree matrix and Indicator matrix
$\mathcal{I}, \mathcal{H}$	Identity matrix and Incidence matrix
N_g	Number of graph convolution layers
Q_a, K_a, V_a	Query, Key, and Value in Attention
N_h, N_d	Number of attention heads, and its dimension
$\mathcal{L}_c, \gamma$	Cauchy loss and its scaling parameter
λ	Outlier ratio
μ, σ_d	Mean and standard deviation
$\bar{\mu}, \bar{\mathcal{N}}$	Mean-centered values, and Normalized standard deviation
c	Grey-sheep detection coverage factor
t	Number of graph convolution matrix factorization layers
$Rel(\cdot), \mathscr{G}_A$	Reliability score and Grey-sheep anomaly score
$\mathscr{G}_D, \mathscr{G}_L$	Grey-sheep discrepancy index and Grey-sheep labeling

ξ	Cold-start percentage
$\mathbb{B}, \kappa$	Poincaré Ball and its curvature
g_E, g_x	Euclidean inner product and Conformal metric
ω	Conformal factor for geometric distortion
$\exp^{\kappa}_{<0>}$	Exponential map at origin
$\log^{\kappa}_{<0>}$	Logarithmic map at origin
ϕ, Θ	SNR and Cross-SNR functions
π, U	Bernoulli distribution and Uniform distribution
z, β, ζ	Coding variable and Stretching factors
τ, δ	Smoothness factor and Thresholds
ρ, Λ, z_α	EMA, Regularization, and Z-score coefficient
φ	Moving average momentum factor
α	Confidence levels or Confidence interval
H_0	Null hypothesis

List of Abbreviations

ADMM	Alternating Direction Method of Multipliers
AE/AI	Auto-Encoder / Artificial Intelligence
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
ARRQP	Anomaly-Resilient Real-time QoS Prediction
AS/ASN	Autonomous System / Autonomous System Number
AV/BN	Autonomous Vehicle / Batch Normalization
CF/CI/CL	Collaborative Filtering / Confidence Interval / Confidence Level
CRRQP	Cold-start Resilient Real-time QoS Prediction
CS/CSM/CSP	Cold-start / Cosine Similarity Metric / Cold-start Percentage
CSS/CSU	Cold-start Service / Cold-start User
CT/Conv1D	Cost / 1D Convolution
CPU	Central Processing Unit
DA/DL	Deep Architectures / Deep Learning
DWA/EqW	Dynamic Weight Averaging / Equal Weighting
EMA	Exponential Moving Average

FCU/FFN First-order Convolution Unit / Feed-Forward Network

FIG First-order user-service Interaction Graph

FLOPs Floating-Point Operations Per Second

FM/FSFB Factorization Machine / Feature Sharing and Fusion Block

GA/GAT Grey-sheep Anomaly / Graph Attention Network

GCMF/GCMFU Graph Convolutional Matrix Factorization / GCMF Unit

GCN/GD Graph Convolution Network / Grey-sheep Detection

GDI/GDM Grey-sheep Discrepancy Index / Grey-sheep Detection Module

GMM/GNN Grey-sheep Mitigation Module / Graph Neural Network

GPAM Global Pattern Adaptation Module

GPU/GPS Graphical Processing Unit / Global Positioning System

gRPC gRPC Remote Procedure Call

GRRQP Grey-sheep Resilient Real-time QoS Prediction

GRU Gated Recurrent Unit

GS/GSS/GSU Grey-sheep / Grey-sheep Services / Grey-sheep Users

HCFM Hypergraph Collaborative Filtering Module

HCN Hypergraph Convolution Network

HCTN Hypergraph Convoluted Transformer Network

HCU Hypergraph Convolution Unit

HFEB Hierarchical Feature Extraction Block

HHGCN Hyperbolic Hypergraph Convolution Network

HUW Heteroscedastic Uncertainty Weighting

HyConv Hyperbolic Convolution

HyGCN Hyperbolic Graph Convolution Network

ID /iForest Identifier / Isolation Forest

IoT/IP/iTree Internet of Things / Internet Protocol / Isolation Tree

JQPM Joint QoS Prediction Module

KKT/LDA Karush–Kuhn–Tucker / Latent Dirichlet Allocation

LM/LN/LOF Language Model / Layer Normalization / Local Outlier Factor

LPAM Local Pattern Adaptation Module

LPEM Localized Pattern Enhancement Module

LSTM/LT Long Short-Term Memory / Latency

MAE/MF Mean Absolute Error / Matrix Factorization

MHA Multi-Head Attention

MhGCMF Multi-head Graph Convolution Matrix Factorization

MhGCMFU Multi-Head GCMF

ML/MLP Machine Learning / Multi-Layer Perceptron

MoE/MSE/MTL Mixture of Expert / Mean Squared Error / Multi-Task Learning

NGS/NMD Non-Greysheep / Non-negative Matrix Decomposition

NMF/OD Non-negative Matrix Factorization / Outlier Detection

QIHG/PARAMS QoS Invocation Hypergraph / Training Parameters

PCC/PG Pearson Correlation Coefficient / Performance Gain

PSO/PTE Particle Swarm Optimization / Predictive Transformer Encoder

PW/QIG/QIV Power / QoS Invocation Graph / QoS Invocation Vector

QoS Quality of Service

RAM/RE/ReLU Random Access Memory / Reliability / Rectified Linear Units

RG/RMSE Region / Root Mean-Squared Error

RNN/RQ/RT Recurrent Neural Network / Research Question / Response Time

SDPA/SGD Scaled Dot-Product Attention / Stochastic Gradient Descent

SHARP-QoS Sparsely-Gated Hierarchical Adaptive Routing for Joint QoS Prediction

SNR/SOA Sub-Network Routing / Service-Oriented Architecture

SORRQP Sparsity & Outlier Resilient Real-time QoS Prediction

SORRTQP Sparsity & Outlier Resilient Real-time Temporal QoS Prediction

SOTA/SSIG State-of-the-art / Second-order Service Interaction Graph

SSCU/STDEV Second-order Service Convolution Unit / Standard Deviation

SUCU/SUIG Second-order User Convolution Unit / Second-order User Interaction Graph

TrD/TeD Training Dataset / Testing Dataset

TE/TF/TP Transformer Encoder / Tensor Factorization / Throughput

TPE Temporal Positional Encoding

TPMCF/TQP Temporal QoS Prediction using Multi-Source Collaborative Features / Temporal QoS Prediction

TS Temporal Smoothing

WSDREAM QoS Benchmark Dataset

Chapter 1

Introduction

Service-oriented computing has evolved into a foundational paradigm for modern digital infrastructures, where geographically distributed services interact with diverse users under continuously changing network environments. In such ecosystems, users are often presented with multiple functionally similar services whose runtime performance varies significantly across locations, time, and workload conditions. As a result, Quality of Service (QoS) prediction has become a core enabling component for reliable service recommendation, selection, and composition.

Unlike traditional recommendation problems that rely primarily on preference signals, QoS prediction concerns measurable performance attributes such as response time, throughput, and reliability. These attributes directly influence system efficiency and user experience, and in many emerging applications, such as cloud computing, edge platforms, and intelligent transportation systems, accurate QoS estimation is closely tied to operational stability and safety. Furthermore, in latency-critical environments, inaccurate predictions may propagate through dependent modules, leading to degraded responsiveness or suboptimal decision-making. Therefore, QoS prediction is not merely a personalization task but a system-level requirement for dependable service orchestration.

Despite extensive research efforts, accurate QoS modeling remains challenging due to the intrinsic properties of real-world service interaction data. First, QoS matrices are highly sparse, as users typically invoke only a small subset of available services.

This sparsity weakens similarity estimation and limits the effectiveness of conventional collaborative filtering methods. The problem becomes more pronounced in cold-start scenarios, where new users or services lack sufficient historical interactions for reliable modeling. Second, QoS observations often suffer from credibility issues. Transient network fluctuations, measurement noise, unstable service behavior, and atypical invocation patterns introduce outliers into the data. In addition, certain users or services exhibit behavior that significantly deviates from global collaborative trends, commonly referred to as grey-sheep phenomena. Such irregularities destabilize learning algorithms and degrade predictive consistency if not explicitly addressed. Third, QoS behavior is inherently dynamic. Service performance evolves over time due to workload variation, network congestion, infrastructure updates, and user mobility. Static models that ignore temporal dependencies fail to capture these evolving patterns. At the same time, user–service interactions exhibit structural dependencies that extend beyond simple pairwise relationships, motivating the need for expressive relational modeling capable of capturing higher-order connectivity. Finally, practical service environments increasingly require the joint consideration of multiple QoS attributes. Modeling response time independently from throughput or reliability neglects their shared underlying factors. However, naïve multi-task learning can introduce negative transfer due to heterogeneous attribute scales and conflicting optimization objectives. Designing a principled framework that supports joint multi-QoS prediction while preserving task-specific characteristics remains a non-trivial challenge.

Although prior studies have explored sparsity mitigation, anomaly handling, representation learning, and temporal modeling individually, these challenges are often treated in isolation. There remains a need for an integrated framework that simultaneously addresses expressive data representation, structural and temporal dependencies, anomaly resilience, and multi-attribute learning in a coherent manner.

Motivated by these limitations, this thesis develops a series of robust and temporally-aware QoS prediction frameworks that systematically tackle sparsity, credibility, and multi-task modeling challenges. The proposed approaches progressively advance from graph-enhanced matrix factorization models to unified hypergraph-

transformer architectures capable of capturing higher-order interactions and temporal evolution while maintaining robustness against anomalous observations. Furthermore, a task-aware multi-QoS learning strategy is introduced to enable stable joint prediction of multiple interdependent performance attributes.

Through this progression, the thesis aims to establish a principled and scalable foundation for reliable QoS prediction in dynamic service ecosystems, supporting accurate, adaptive, and dependable service selection across both large-scale online platforms and real-world application domains.

1.1 Motivation of the Thesis

The rapid expansion of digital service ecosystems has resulted in an unprecedented growth in both the number of available services and the scale of user interactions, significantly intensifying the demand for accurate, personalized, and dependable service recommendation mechanisms. In domains such as e-commerce, cloud platforms, and service marketplaces, users are often presented with a multitude of functionally similar services whose actual runtime performance differs substantially due to variations in network conditions, deployment environments, and user-specific factors. Consequently, reliable Quality of Service (QoS) estimation has become a critical prerequisite for effective service recommendation, selection, and composition, directly influencing user satisfaction and operational efficiency.

Beyond traditional e-commerce platforms, service-oriented computing has become deeply embedded in latency-sensitive and mission-critical real-world applications, including smart transportation systems, edge-enabled Internet of Things (IoT), and autonomous vehicle (AV) ecosystems. Such systems rely on the seamless orchestration of multiple concurrent services supporting tasks such as perception, localization, prediction, planning, and control. Each of these tasks is often backed by distinct services whose QoS characteristics, e.g., response time, throughput, and reliability, must be estimated accurately and in real-time. In these environments, erroneous or delayed QoS predictions can propagate across the decision pipeline, adversely affecting

system responsiveness, reliability, and safety, thereby elevating QoS prediction from a recommendation problem to a core system-dependability concern.

Accurate QoS prediction in such settings is fundamentally challenged by the characteristics of real-world QoS data. First, QoS observations are extremely sparse due to limited user–service interactions and the frequent introduction of new users and services. Second, QoS measurements are often contaminated by anomalies arising from transient network fluctuations, measurement noise, outliers, and atypical invocation patterns exhibited by grey-sheep users or services. Third, QoS behavior is shaped by complex structural dependencies involving users, services, temporal dynamics, and contextual factors such as geographical location and network topology. These intertwined factors severely limit the effectiveness of conventional collaborative filtering and static modeling approaches.

Moreover, modern service-oriented applications increasingly require the joint consideration of multiple QoS parameters to assess overall service suitability. For example, optimizing response time alone is insufficient if throughput, reliability, or availability are simultaneously degraded. Existing single-metric or naïvely shared multi-task models often fail to capture inter-QoS dependencies and frequently suffer from negative transfer due to heterogeneous numerical scales and conflicting optimization objectives. This limitation is particularly pronounced in real-time and safety-critical systems, where consistent and balanced multi-QoS estimation is essential for dependable operation.

Motivated by these challenges and practical deployment requirements, this thesis aims to develop reliable, anomaly-resilient, and real-time QoS prediction frameworks that systematically address data sparsity, data credibility, expressive representation learning, temporal dynamics, and joint multi-QoS optimization. The proposed approaches leverage advanced graph-based, hypergraph-based, and deep temporal modeling techniques to capture higher-order user–service interactions while maintaining robustness against anomalies and high responsiveness. Ultimately, this research seeks to enable precise, adaptive, and trustworthy service selection across both large-scale online platforms and emerging real-world application domains, including smart trans-

portation and autonomous systems.

1.2 Problem Statement

Service-oriented computing environments have evolved into highly dynamic, heterogeneous, and large-scale ecosystems, where geographically distributed users interact with numerous services under continuously varying network and system conditions. In such environments, Quality of Service (QoS) prediction plays a critical role in enabling reliable service selection, recommendation, and autonomous decision-making by estimating service performance prior to invocation. However, accurate QoS prediction remains fundamentally challenging due to the intrinsic characteristics of real-world service interaction data, including extreme sparsity, complex structural dependencies, temporal variability, and credibility issues arising from noisy and anomalous observations. These factors significantly limit the effectiveness, robustness, and generalization capability of conventional QoS prediction methods.

First, QoS prediction requires expressive data representation capable of capturing intricate user–service interaction patterns. Traditional matrix factorization and similarity-based methods primarily model pairwise relationships and often overlook higher-order dependencies and temporal evolution. In dynamic service ecosystems, QoS values fluctuate over time due to network variability, workload changes, and user mobility. Consequently, constructing spatio-temporal representations that adequately reflect both structural and temporal dependencies remains a core modeling challenge.

Second, real-world QoS datasets are highly sparse. Users typically invoke only a small subset of available services, resulting in incomplete interaction matrices. Severe sparsity weakens similarity estimation, limits collaborative filtering effectiveness, and exacerbates the cold-start problem for new users and services. Ensuring reliable prediction under such data-deficient conditions requires modeling strategies that can infer meaningful structure from limited observations.

Third, beyond sparsity, QoS data often suffers from credibility issues. Observed QoS values may contain outliers due to unstable network conditions, measurement

noise, or unreliable service behavior. Additionally, certain users or services exhibit distinctive patterns that do not conform to global collaborative trends, commonly referred to as grey-sheep behavior. These factors introduce instability into learning algorithms and degrade prediction reliability. Therefore, anomaly resilience and data credibility modeling are not auxiliary concerns but fundamental requirements for dependable QoS prediction.

Finally, most existing approaches treat QoS attributes independently, modeling response time, throughput, and other performance metrics in isolation. In practice, these attributes are often interdependent and influenced by shared underlying factors. While joint modeling offers the potential to exploit shared representations, it introduces additional complexity, including negative transfer across tasks and numerical instability due to differing attribute scales. Designing a principled multi-task framework that enables joint prediction while preserving task-specific characteristics remains an open research problem.

Although prior studies have addressed aspects of representation learning, sparsity mitigation, anomaly handling, or multi-attribute prediction individually, these challenges are rarely considered in a unified manner. There remains a lack of an integrated, temporally-aware, and credibility-resilient framework that simultaneously captures complex spatio-temporal dependencies, alleviates sparsity, accounts for anomalous behavior, and supports joint prediction of multiple QoS attributes for personalized service environments.

Addressing this gap forms the central research problem of this thesis.

1.3 Objectives of the Thesis

The overarching objective of this thesis is to develop reliable, and real-time Quality of Service (QoS) prediction frameworks by systematically addressing the fundamental challenges inherent in QoS modeling for service-oriented computing environments. Specifically, this research pursues the following objectives:

1. **Multi-Source Data Representation:** To design and learn expressive QoS rep-

resentations by integrating domain-specific attributes with automatically learned higher-order latent features, thereby enhancing the modeling capacity and predictive accuracy in dynamic service environments.

2. **Sparsity Mitigation and Anomaly Resilience:** To alleviate the impact of data sparsity through graph-based relational modeling, while simultaneously improving robustness against anomalous QoS observations caused by noisy users, unreliable services, or fluctuating network conditions.

3. **Unified Modeling for Data Representation, Sparsity Handling, and Anomaly Resilience:** To develop a cohesive and temporally-aware framework that jointly addresses feature representation, sparsity, and anomaly handling, enabling consistent and stable QoS prediction under evolving service dynamics.

4. **Joint QoS Prediction:** To formulate a unified multi-task learning framework that supports the simultaneous prediction of multiple interdependent QoS attributes, while mitigating negative transfer and preserving task-specific discriminative characteristics.

1.4 Contributions of the Thesis

This thesis makes the following primary contributions to the field of QoS prediction in service-oriented computing:

1. **Advanced Data Representation Learning:** We propose a collaborative spatio-temporal modeling framework that integrates domain-specific QoS features with automatically learned higher-order latent representations. By combining graph convolutional matrix factorization with a predictive transformer encoder, the proposed framework captures both spatial dependencies among user–service interactions and their temporal evolution. This design enhances representation expressiveness and improves prediction accuracy in dynamic service environments.

2. **Robust QoS Prediction Learning under Sparsity and Anomalies:** To address the practical challenges of data sparsity, and anomalous QoS observations, we introduce a multi-head graph convolution matrix factorization architecture with residual learning and an outlier-resilient loss formulation. The proposed framework incorporates dedicated mechanisms for grey-sheep detection and cold-start handling, thereby improving robustness and stability in static QoS prediction scenarios.

3. **Unified Framework for Data Representation, Sparsity, and Credibility:** Building upon the previous studies, we develop the Hypergraph Convoluted Transformer Network (HCTN), a unified architecture that jointly models higher-order collaborative relationships, temporal dynamics, and data credibility issues. By integrating hypergraph learning, dual-channel transformer networks, and credibility-aware learning strategies, the framework effectively captures complex triadic interactions among users, services, and time, leading to enhanced predictive robustness.

4. **Multi-QoS Prediction with Task-Aware Learning:** Moving beyond single-metric modeling, we propose a multi-task QoS prediction framework that enables simultaneous estimation of multiple interdependent QoS attributes. The framework incorporates hierarchical feature sharing and an exponential moving average (EMA)-based loss balancing strategy to mitigate negative transfer across tasks. This approach improves convergence stability and generalization performance in multi-QoS prediction settings.

5. **Extensive Empirical Validation across Benchmark and Real-World Datasets:** Comprehensive experimental studies are conducted on multiple benchmark datasets, including WSDREAM variants and real-world QoS datasets with multiple attributes. The results consistently demonstrate the effectiveness, robustness, and real-time responsiveness of the proposed frameworks under varying sparsity levels, anomaly ratios, and multi-task configurations.

1.5 Organization of the Thesis

The remainder of this thesis is organized as follows.

- Chapter 2 provides the necessary background and presents a comprehensive review of the related literature, along with an overview of the benchmark web service datasets and the evaluation metrics used in this thesis.
  - A survey article related to this chapter is currently under preparation.
- Chapter 3 introduces our proposed solution to address the data representation problem in dynamic service environments. This chapter is derived from:
  - Suraj Kumar, Soumi Chattopadhyay and Chandranath Adak, "TPMCF: Temporal QoS Prediction Using Multi-Source Collaborative Features," in IEEE Transactions on Network and Service Management, vol. 21, no. 4, pp. 3945-3955, Aug. 2024, doi: 10.1109/TNSM.2024.3395428.
- Chapter 4 proposes an anomaly-resilient framework for static QoS prediction to mitigate the adverse effects of data sparsity and anomalous QoS observations. This chapter is derived from:
  - Suraj Kumar and Soumi Chattopadhyay, "ARRQP: Anomaly Resilient Real-Time QoS Prediction Framework With Graph Convolution," in IEEE Transactions on Services Computing, vol. 18, no. 3, pp. 1245-1261, May-June 2025, doi: 10.1109/TSC.2025.3565376.
- Chapter 5 presents a unified framework that jointly addresses data representation, sparsity, and anomaly handling in time-aware QoS modeling. This chapter is derived from:
  - Suraj Kumar, Soumi Chattopadhyay and Chandranath Adak, "Anomaly-Resilient Temporal QoS Prediction Using Hypergraph Convoluted Transformer Network," in IEEE Transactions on Network and Service Management, vol. 23, pp. 3556-3568, 2026, doi: 10.1109/TNSM.2026.3674650.

- Chapter 6 presents a joint QoS prediction framework that extends traditional single-metric approaches to support the simultaneous prediction of multiple interdependent QoS parameters. This chapter is derived from:

  – Suraj Kumar, Arvind Kumar, Soumi Chattopadhyay, "SHARP-QoS: Sparsely-gated Hierarchical Adaptive Routing for Joint Prediction of QoS", IEEE Transactions on Services Computing. https://arxiv.org/pdf/2512.17262.(Under Review)

- Finally, Chapter 7 concludes the thesis by summarizing the key findings and outlining promising future research directions.

Chapter 2

Background and Literature Survey

In this chapter, we present a brief overview of the literature study with a few background concepts related to the QoS prediction, underlying challenges and state-of-the-art (SOTA) solutions.

2.1 Background and Preliminaries

To establish the foundation of this work, we first introduce key concepts related to web services, QoS parameters, QoS prediction paradigms, and the fundamental challenges associated with QoS modeling.

2.1.1 Web Services and QoS Parameters

Definition 1 (Web Service)**.** *A web service is a reusable software component that performs a specific task through machine-to-machine interactions, thereby bridging business processes and IT services.* ■

Traditional web services are typically described using the Web Service Description Language (WSDL), commonly expressed in XML format, which specifies service interfaces, communication protocols, and network endpoints [2]. Service identification is achieved through a Uniform Resource Identifier (URI), which uniquely identifies logical or physical resources using standardized internet protocols [3]. These services

expose their functionality through self-describing XML-based interfaces built upon widely adopted internet standards [4]. In contrast to this, modern service-oriented architectures increasingly adopt high-performance communication frameworks such as gRPC [1], which use Protocol Buffers to define service interfaces and message structures.

Web services are generally characterized by two categories of parameters: *(i) functional parameters*, which define inputs, outputs, and core functionality; and *(ii) non-functional parameters*, commonly referred to as *Quality of Service (QoS) parameters.*

Definition 2 (QoS Parameter (q))**.** *A QoS parameter is a non-functional attribute used to measure and evaluate the performance of a web service.* ■

Notably, while multiple web services may offer similar functionality, they can be distinguished by their QoS characteristics. Representative QoS parameters [5] are summarized in Table 2.1.

In globally distributed service-oriented architectures (SOA) [3], QoS plays a critical role in determining service performance. Although service providers typically specify QoS values prior to deployment, these values may vary significantly in practice. Consequently, accurately assessing and selecting optimal services becomes challenging, motivating the need for QoS prediction techniques.

2.1.2 QoS Prediction Paradigms

QoS prediction can be formulated as a missing-value estimation problem. It involves forecasting service performance to enable informed decision-making by users and au-

[1] https://grpc.io/

Table 2.1: Examples of QoS parameters

QoS Parameter	Units	Optimization Objective	Definition
Response Time	millisecond (ms)	Minimize	Time interval between sending a request and receiving a response
Latency	millisecond (ms)	Minimize	Time taken by the server to process a given request
Throughput	Kilobits per second (kbps)	Maximize	Data transmission rate of a service over a network within a given time frame
Reliability	%	Maximize	Probability of failure-free execution over time.
Power	Watt (W)	Minimize	Power consumption of the service during execution
Cost	Monetary unit	Minimize	Service usage cost incurred per invocation
Availability	%	Maximize	Proportion of time the service is operational
Jitter	millisecond (ms)	Minimize	Variation in packet delay over time

tomated systems.

Definition 3 (QoS Prediction)**.** *Given a web service environment with a set of users* $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ *and a set of services* $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$*, along with a QoS interaction matrix* $\mathcal{Q} \in \mathbb{R}^{n \times m}$*, the objective of QoS prediction is to estimate the missing values* $q_{i,j} = 0$ *for target user–service pairs* (u_i, s_j)*.* ■

QoS prediction is fundamental to several stages of the service life cycle. For example, service recommendation [6–8] and service composition [9, 10] rely heavily on accurate QoS estimation to support informed service selection. Based upon the requirements, the different context have been followed in the literature for QoS prediction.

Definition 4 (Static QoS Prediction)**.** *Let* $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ *be a set of n users, each associated with contextual information* $\mathcal{C}^u$ *(e.g., user ID, region, autonomous system), and* $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ *a set of m web services, each with contextual data* $\mathcal{C}^s$ *(e.g., service ID, region, provider). Given a QoS parameter q, we define a QoS log matrix* $\mathcal{Q} \in \mathbb{R}^{n \times m}$*, where each non-zero entry* q_{ij} *represents the observed value of q for service* s_j *invoked by user* u_i*, and zero entries indicate missing interactions:*

$$\mathcal{Q} = \begin{cases} q_{ij} \in \mathbb{R}_{>0}, & \text{value of } q \text{ of } s_j \text{ invoked by } u_i \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The matrix $\mathcal{Q}$ *is typically sparse. The goal of QoS prediction is to accurately estimate the missing values* $q_{ij} = 0$ *for target user-service pairs, thereby minimizing the prediction error.* ■

However, in practice, QoS values often fluctuate over time due to network congestion, concurrent access, and application behavior [11]. This dynamic nature necessitates continuous monitoring and adaptive prediction mechanisms. The temporal QoS prediction problem is mathematically formulated as follows.

Definition 5 (Temporal QoS Prediction)**.** *Given a set of users* $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$*, a set of services* $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$*, a QoS parameter q, and historical observations*

over T time-steps, the QoS invocation log is represented as a third-order tensor $\mathcal{Q} \in \mathbb{R}_{+}^{n \times m \times T}$, defined as:

$$\mathcal{Q}(i,j,t) = \begin{cases} q_{ij}^{t}, & \textit{if user } u_i \textit{ invokes service } s_j \textit{ at time } t, \\ 0, & \textit{otherwise.} \end{cases} \tag{2.2}$$

The objective of temporal QoS prediction is to estimate the unknown QoS value q_{ij}^{T+1} for a target user–service pair $(u_i, s_j) \in \mathcal{U} \times \mathcal{S}$ using the historical observations contained in $\mathcal{Q}$. ■

Since service environments are often subject to anomalies (e.g., outliers), QoS prediction should account for such irregularities to ensure reliable estimation.

Definition 6 (Credibility-aware QoS Prediction)**.** *Given users $\mathcal{U}$ and services $\mathcal{S}$, credibility-aware QoS prediction aims to estimate missing QoS values while accounting for anomalous observations within the interaction matrix.* ■

In many practical scenarios, service selection depends on multiple QoS parameters rather than a single attribute, which motivates the need for joint modeling.

Definition 7 (Multi-QoS Prediction)**.** *Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ denote the set of n users, and $\mathcal{S} = \{s_1, \ldots, s_m\}$ the set of m services, with each associated with their context information, including the geographical region (RG) and autonomous system (AS).*

Assume $\mathcal{Q} = \{\mathcal{Q}^1, \mathcal{Q}^2, \cdots, \mathcal{Q}^P\}$ is the set of P QoS parameters, where each $\mathcal{Q}^p \in \mathbb{R}^{n \times m}$ denotes partially observed user–service interactions matrix with entries

$$\mathcal{Q}_{ij}^{p} = \begin{cases} q_{ij}^{p} > 0, & \textit{if user } u_i \textit{ invokes service } s_j, \\ 0, & \textit{otherwise.} \end{cases} \tag{2.3}$$

The objective of this formulation is to design a joint QoS prediction framework that learns a single model to predict all P *QoS parameters simultaneously.*

■

2.1.3 Characteristics and Objectives

Understanding the intrinsic characteristics of web services is essential for designing effective QoS prediction frameworks. These characteristics directly influence modeling choices, robustness requirements, and evaluation strategies.

Definition 8 (Web Service Characteristics)**.** *Web services exhibit several inherent properties that influence QoS prediction, including: (i) heterogeneity of users and services, (ii) dynamic and time-varying QoS behavior, (iii) sparsity of user–service interactions, (iv) geographical and network dependency, and (v) susceptibility to anomalous observations such as outliers and irregular usage patterns.* ■

These characteristics collectively make QoS prediction a challenging task, requiring models that are robust, scalable, and capable of adapting to evolving service environments. In addition to understanding these characteristics, it is important to clearly define the overarching objectives of QoS prediction within service-oriented computing systems.

Definition 9 (Objectives)**.** *The primary objective of QoS prediction is to accurately and efficiently estimate missing or future QoS values in order to support reliable service recommendation, selection, and composition. This further entails ensuring robustness to anomalies, scalability to large service ecosystems, adaptability to temporal variations, and responsiveness suitable for real-time applications.* ■

Together, these objectives guide the development of the proposed frameworks in this thesis, which aim to balance predictive accuracy, robustness, scalability, and practical deployability.

2.1.4 Various Modeling Paradigm of QoS Prediction Problem

In real-world environments, the QoS value observed when a user invokes a service is not a static or deterministic parameter. Instead, it varies across multiple dimensions, including users [6], geographical locations [12, 13], temporal conditions [14], and trust-related factors associated with users or services [15, 16]. These variations make

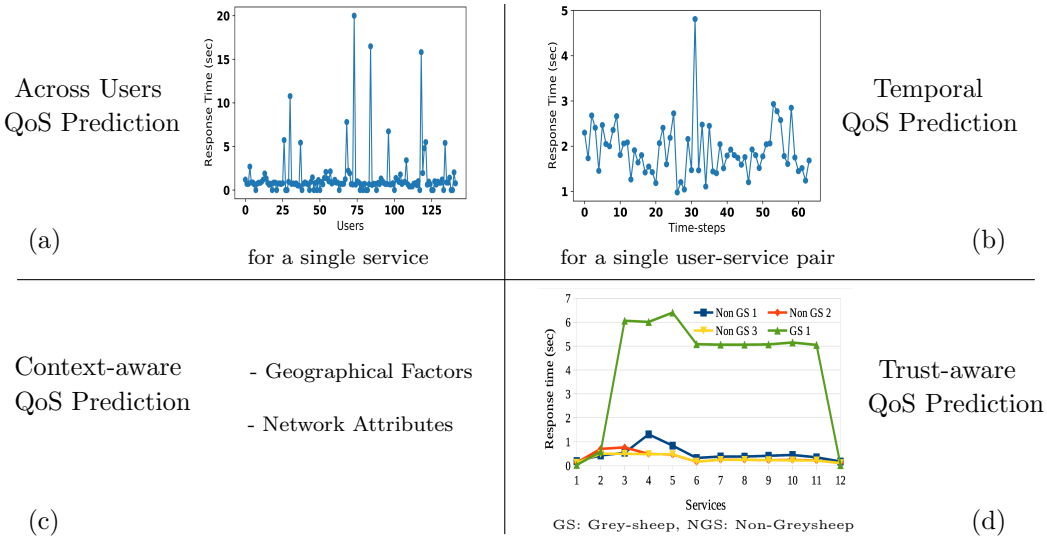


Figure 2.1: Different modeling perspectives of the QoS prediction problem.

QoS prediction a non-trivial task and requires careful problem modeling. Since QoS prediction plays a fundamental role in service-oriented computing and applications such as service selection, recommendation, and composition, an appropriate modeling strategy becomes critical. In the following subsections, we discuss the major factors influencing QoS prediction, as illustrated in Fig. 2.1.

2.1.4.1 QoS Prediction Across Users

As defined in Definition 3, the QoS value of a given service may differ significantly across users [8]. This variation primarily arises due to heterogeneous network conditions, user-side infrastructure, and geographical distribution. As shown in Fig. 2.1(a), the response time of a single service varies when invoked by different users. This observation confirms that user-specific characteristics play a crucial role in QoS modeling. Therefore, user heterogeneity must be carefully incorporated into QoS prediction models to achieve accurate estimations.

2.1.4.2 Temporal QoS Prediction

In practical scenarios, even for a fixed user–service pair, QoS values are not constant over time. Continuous fluctuations in network traffic, server load, bandwidth avail-

ability, and possible network failures cause temporal variations in QoS observations. As illustrated in Fig. 2.1(b), the QoS values for a single user–service pair change across different timestamps. Consequently, the objective of temporal QoS prediction is to estimate the QoS value of a target service for a given user at a specific time with high precision [17–20]. Therefore, incorporating temporal dynamics into the QoS modeling is critical.

2.1.4.3 Context-Aware QoS Prediction

QoS values are also influenced by contextual attributes associated with users and services. These contextual factors include geographical information (e.g., region, latitude–longitude coordinates), network-level characteristics (e.g., autonomous systems, IP domains), and other environmental conditions. As depicted in Fig. 2.1(c), QoS values fluctuate across different contextual settings. Ignoring such contextual information may lead to suboptimal predictions. Therefore, context-aware QoS prediction models aim to explicitly integrate spatial and network-related attributes into the learning framework [12, 13, 21].

2.1.4.4 Trust-Aware QoS Prediction

Collaborative QoS prediction mechanisms rely heavily on the assumption that user observations are reliable. However, in practice, the presence of untrustworthy, malicious, or atypical users can significantly distort prediction performance. As shown in Fig. 2.1(d), atypical users (often referred to as grey-sheep (GS) users) exhibit invocation patterns that differ considerably from normal or non-greysheep (NGS) users. Such behavior introduces credibility and reliability challenges in collaborative modeling. Therefore, trust-aware QoS prediction methods aim to identify and mitigate the influence of unreliable observations to ensure robust and dependable predictions [22].

2.2 Literature Review

In this section, we present a comprehensive review of prior research on QoS prediction. The existing methods were systematically classified according to collaborative filtering (CF) paradigms, and their limitations were critically examined across multiple modeling dimensions. In addition, we explored multi-QoS prediction approaches that jointly estimate multiple QoS parameters, discussing both their methodological advancements and open research challenges.

2.2.1 Collaborative Filtering Paradigm for QoS Prediction

To address the QoS prediction problem, numerous approaches have been proposed in the literature, most of which adopt collaborative filtering (CF) strategies [11, 23]. As illustrated in Fig. 2.2, the CF paradigm can be broadly categorized into three classes: (i) memory-based (or neighborhood-based) methods, (ii) model-based methods, and (iii) hybrid approaches. In the following subsections, we discuss each category and review representative techniques.

2.2.1.1 Memory-Based or Neighborhood-Based CF for QoS Prediction

Early QoS prediction methods predominantly relied on memory-based CF techniques, also referred to as neighborhood-based methods. The fundamental assumption underlying these approaches is that users with similar historical QoS experiences are likely to observe similar QoS values in the future. Accordingly, memory-based methods identify similar users based on QoS interaction data using similarity metrics such as cosine similarity measure (CSM), Pearson Correlation Coefficient (PCC), and ratio-

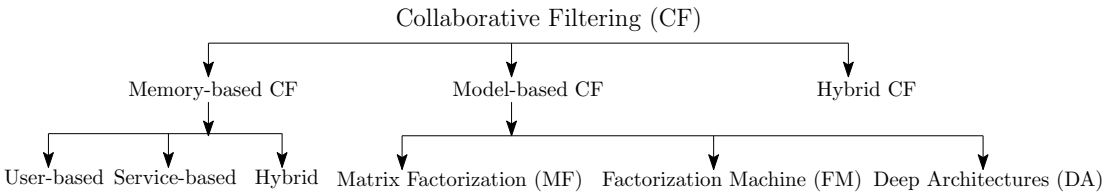


Figure 2.2: Types of CF-based approaches for QoS prediction.

based similarity measures. Furthermore, geographical proximity has been incorporated using the Haversine distance (HD) to enhance similarity estimation. Each similarity metric is defined below.

Definition 10 (Cosine Similarity Measure (CSM))**.** *Given two users QoS vectors* $\mathcal{Q}_i, \mathcal{Q}_j \in \mathbb{R}^m$*, the cosine similarity between them, denoted as* $CSM(\mathcal{Q}_i, \mathcal{Q}_j)$*, is defined as:*

$$CSM(\mathcal{Q}_i, \mathcal{Q}_j) = \frac{\mathcal{Q}_i \cdot \mathcal{Q}_j}{||\mathcal{Q}_i||\ ||\mathcal{Q}_j||} = \frac{\sum_{k=1}^{n} \mathcal{Q}_i(k)\mathcal{Q}_j(k)}{\sqrt{\sum_{k=1}^{n} \mathcal{Q}_i^2(k)}\sqrt{\sum_{k=1}^{n} \mathcal{Q}_j^2(k)}}. \tag{2.4}$$

■

Definition 11 (Pearson Correlation Coefficient (PCC))**.** *Given two m dimensional QoS vectors* $\mathcal{Q}_i, \mathcal{Q}_j \in \mathbb{R}^m$*,* $PCC(\mathcal{Q}_i, \mathcal{Q}_j)$ *is defined as follows:*

$$PCC(\mathcal{Q}_i, \mathcal{Q}_j) = \frac{\sum_{k=1}^{m} (\mathcal{Q}_i(k) - \bar{\mathcal{Q}}_i)(\mathcal{Q}_j(k) - \bar{\mathcal{Q}}_j)}{\sqrt{\sum_{k=1}^{m} (\mathcal{Q}_i(k) - \bar{\mathcal{Q}}_i)^2}\sqrt{\sum_{k=1}^{m} (\mathcal{Q}_j(k) - \bar{\mathcal{Q}}_j)^2}} \tag{2.5}$$

where, $\bar{\mathcal{Q}}_i$ *and* $\bar{\mathcal{Q}}_j$ *are the mean of* $\mathcal{Q}_i$ *and* $\mathcal{Q}_j$*, respectively.* ■

It may be noted that PCC measures the linear correlation between two vectors. It is the ratio between the covariance of V_i and V_j and the product of their standard deviations. PCC is a normalized covariance measure; hence, the value ranges from -1 to 1. Here also, V_i and V_j are either the QoS vectors of two users or two services [8].

Definition 12 (Ratio-based Similarity Measure (RBS))**.** *Given two QoS vectors* $\mathcal{Q}_i, \mathcal{Q}_j \in \mathbb{R}^m$*, the ratio-based similarity measure [24] is defined as:*

$$RBS(\mathcal{Q}_i, \mathcal{Q}_j) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \frac{\min(\mathcal{Q}_i(k), \mathcal{Q}_j(k))}{\max(\mathcal{Q}_i(k), \mathcal{Q}_j(k))} \tag{2.6}$$

where,

$$\mathcal{K} = \{k \mid \mathcal{Q}_i(k) \neq 0 \text{ and } \mathcal{Q}_j(k) \neq 0\}$$

denotes the set of indices where both QoS vectors have valid observations. ■

$\mathcal{Q}_i$ and $\mathcal{Q}_j$, in the above definition, refer to the QoS invocation sets (i.e., the QoS invocation set of a user u_j consists of services invoked by u_i; similarly, the QoS invocation set of a service s_j comprises users invoked s_j) of two users/services in our context [25].

While the above similarity metrics quantify relationships between QoS vectors, the Haversine distance (HD) is used to measure geographical proximity between users or services based on their location coordinates [26, 27].

Definition 13 (Haversine Distance (HD))**.** *Given the geographical coordinates of two points on the Earth's surface, $\gamma_i = (\phi_i, \psi_i)$ and $\gamma_j = (\phi_j, \psi_j)$, where ϕ and ψ denote latitude and longitude in radians, respectively, the Haversine distance between them is defined as:*

$$HD(\gamma_i, \gamma_j) = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_j - \phi_i}{2}\right) + \cos(\phi_i)\cos(\phi_j)\sin^2\left(\frac{\psi_j - \psi_i}{2}\right)}\right) \quad (2.7)$$

where, r denotes the Earth's radius ($r \approx 6371$ km), and $\arcsin(\cdot)$ represents the inverse sine function. ■

The QoS value of a target user–service pair is estimated through similarity-weighted aggregation of QoS values from the selected neighbors. Depending on how neighbors are identified and utilized, memory-based collaborative filtering methods can be broadly classified into three categories. First, **user-based CF** [6] estimates the QoS value of a target user–service pair by aggregating QoS values from similar users. Second, **service-based CF** [7] follows an analogous strategy by identifying similar services and aggregating their QoS values for the target user. Third, **hybrid CF** approaches [8] combine both user-based and service-based similarities to mitigate user or service bias in QoS estimation. Several representative methods have been proposed under this paradigm. NRCF [28] employs linear interpolation between user-side and service-side QoS predictions. RACF [29] introduces a ratio-based similarity measure tailored for QoS modeling. RECF [24] integrates Pearson Correlation Coefficient (PCC) for user similarity and ratio-based similarity for service similarity, demonstrating improved performance. BRACF [30] addresses the large-range variation problem caused by imbalanced neighbor contributions in user-based CF settings, leading to more stable predictions.

Limitation of Memory-based QoS Prediction. Despite their simplicity, interpretability, and ease of implementation, memory-based approaches suffer from several limitations, including capturing global structure of the data. These include severe data sparsity, vulnerability to noisy or anomalous observations, and limited capability to capture non-linear and higher-order relationships in QoS data. Furthermore, computing pairwise similarities across large-scale datasets introduces significant computational overhead, making these methods less scalable and low responsiveness. A detailed discussion of these challenges is provided in Sec. 2.2.2.

2.2.1.2 Model-based CF for QoS Prediction

To overcome the limitations of memory-based methods, more sophisticated model-based approaches have been developed, which often leverages machine learning and deep learning techniques. The model-based approaches typically employ the reconstruction-based predictive models by constructing a global model using sparse QoS settings, thereby estimating an overall structure of the data. As shown in Fig. 2.2, representative model-based CF types include: (i) Clustering-based approaches, (ii) Matrix factorization (MF) techniques, (iii) Factorization machines (FM) methods, and (iv) Deep neural networks based practices. In the following subsections, we reviewed the existing methods on these approaches in detail.

2.2.1.3 Model-Based CF: Clustering-Based Approaches

Clustering-based approaches constitute another category of model-based collaborative filtering methods for QoS prediction. These techniques group users or services into clusters based on observed QoS values and/or contextual attributes (e.g., geographical or network information). The primary objective is to exploit intra-cluster similarity while reducing computational complexity. In general, clustering algorithms such as K-means are employed to partition users or services into homogeneous groups. The clustering results are then integrated into the QoS prediction framework. For instance, K-means clustering has been adopted in [15, 16] to group similar users and subsequently refine memory-based CF predictions using cluster-level information. By

operating within clusters, these methods reduce computational overhead and mitigate the impact of unreliable observations. Moreover, clustering strategies have also been incorporated into regression-based QoS prediction frameworks. In [26, 31], users or services are clustered according to contextual attributes, and regression models are trained with information from the clustered user or services. Such a design enables the model to capture localized patterns and contextual heterogeneity more effectively than global models. In summary, clustering-based approaches improve scalability and modeling flexibility. However, their performance largely depends on the quality of cluster formation, and inappropriate clustering may propagate errors into subsequent prediction stages.

2.2.1.4 Model-Based CF: Matrix Factorization Approaches

Matrix Factorization (MF) [32] is one of the most widely adopted model-based collaborative filtering techniques. It has been extensively applied in recommender systems, dimensionality reduction, topic modeling, and QoS prediction due to its ability to capture latent interactions between entities. To employ the MF in the QoS prediction context, consider an environment with n users and m services. Let $\mathcal{Q} \in \mathbb{R}^{n \times m}$ denote the partially observed QoS matrix, where each entry q_{ij} represents the observed QoS value when user u_i invokes service s_j. Since $\mathcal{Q}$ is typically sparse, the goal of MF is to approximate it by decomposing it into two low-rank latent factor matrices: $U \in \mathbb{R}^{n \times d}$ and $S \in \mathbb{R}^{m \times d}$, where $d \ll \min(n, m)$ denotes the latent dimension. Each row U_i represents the latent feature vector of user u_i, and each row S_j represents the latent feature vector of service s_j. The QoS value is then approximated as:

$$q_{ij} \approx U_i S_j^T. \tag{2.8}$$

To learn the latent factors, the mean-squared error (MSE)-based objective function is minimized over the observed entries:

$$\min_{U,S} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} \left(\mathcal{Q}_{ij} - U_i S_j^T \right)^2, \qquad \text{where, } I_{ij} = \begin{cases} 1, & \text{if } q_{ij} \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

However, directly minimizing Eq. 2.9 may lead to overfitting, especially under high sparsity conditions. To improve generalization, L2 regularization terms are introduced on the latent factors, resulting in the regularized optimization problem:

$$\min_{U,S} L(U,S) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\left(\mathcal{Q}_{ij} - U_i S_j^T\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2, \tag{2.10}$$

where, $\|\cdot\|_F$ denotes the Frobenius norm, and λ_U, $\lambda_S > 0$ are regularization hyperparameters controlling model complexity.

The objective in Eq. 2.10 corresponds to minimizing the regularized squared reconstruction error over observed entries. The optimization is typically performed using Stochastic Gradient Descent (SGD) [33] or Alternating Least Squares (ALS) [34], iteratively updating U and S until convergence.

Several studies have extended MF for QoS prediction. Early works such as Non-negative Matrix Factorization (NMF) [32] introduced non-negativity constraints to better model QoS data, while Probabilistic Matrix Factorization (PMF) [35] incorporated a probabilistic framework for latent factor learning. To enhance prediction accuracy, neighborhood information was integrated into MF models. Lo et al. [36] designed user- and service-specific regularization terms based on similarity relationships. Zheng et al. [37] further proposed a neighborhood-integrated MF approach that first identifies top-k similar neighbors and then performs MF within the refined neighborhood. However, these methods lack the ability to incorporate user and service context to improve MF performance. Qi et al. [38] combined memory-based and MF approaches by leveraging user-side network attributes such as IP, ASN, and country information. CSMF [39] and GeoMF [12] proposed context-sensitive MF incorporated geographical location in an integrated approach. Xu et al. [40] and Yin et al. [41] extended MF by incorporating geographical and network location information from user and service sides, which also helped alleviate the cold-start problem. CMF [42] employed a robust MF framework for QoS prediction. Beyond matrix-based modeling, Tensor Factorization (TF) methods extend MF to higher-order settings by modeling the QoS data as a three-dimensional tensor (user–service–time). TF-based approaches learn low-rank latent representations for users, services, and temporal dynamics simul-

taneously [42, 43].

2.2.1.5 Model-Based CF: Factorization Machines Approaches

Factorization Machines (FM) are model-based approaches that combine the strengths of polynomial regression and latent factor models. Unlike traditional MF, FM can naturally incorporate additional contextual features while modeling pairwise feature interactions. An order-2 FM model is defined as:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j \sum_{f=1}^{k} v_{i,f} v_{j,f}, \tag{2.11}$$

where, $w_0 \in \mathbb{R}$ is the global bias, w_i denotes the weight of the i-th feature, and $v \in \mathbb{R}^{n \times k}$ represents the latent factor matrix used to model second-order feature interactions.

Several FM-based QoS prediction methods have been proposed. Li et al. [44] introduced an attention-based FM framework that filters invalid services and incorporates contextual information to model user–service correlations. Yang et al. [27] further enhanced similarity modeling by leveraging geographical information and employing the Haversine distance to improve context-aware QoS prediction. To enhance expressivity, Wu et al. [45] utilized embedding techniques to represent users and services via one-hot encoding, enabling feature extraction through a neural network before feeding representations into the FM model. Shen et al. [46] proposed a context-aware deep FM framework that formulates QoS prediction as a regression problem and captures non-linear and higher-order feature interactions using deep architectures. Compared to MF and memory-based methods, FM models effectively capture pairwise feature interactions and integrate contextual information in linear time complexity. However, FM approaches still require careful parameter tuning, depend on predefined feature inputs, and are primarily limited to second-order feature interactions unless extended with deep architectures.

2.2.1.6 Model-Based CF: Deep Architecture-Based Approaches

Deep architecture (DA)-based models employ neural networks to learn non-linear representations for QoS prediction. Compared to conventional MF and FM methods,

these models better capture complex feature interactions and improve predictive accuracy. Early DA-based approaches primarily relied on multi-layer perceptrons (MLPs) to model latent interactions. Zou et al. [47] integrated user–service similarity and AS-based contextual features into an end-to-end MLP framework. Yin et al. [48] enhanced robustness through a data augmentation strategy that generates virtual users using a newly defined similarity measure and trains multiple MLP and MF models. To better exploit contextual information, Yin et al. [49] applied fuzzy clustering on geographical attributes to obtain membership representations, refined them using denoising autoencoders to remove redundancy, and utilized an MLP for final QoS estimation. Similarly, Gao et al. [50] combined fuzzy clustering with a neural collaborative framework to capture both local and global feature interactions. Subsequent studies focused on integrating MF with deep neural networks to jointly model linear and higher-order interactions. NDMF [47], MM-DNN [51], and DCLG [52] combine dot-product-based linear correlations from MF with non-linear representations learned through MLPs, enabling richer interaction modeling. GFEN [19] incorporate the probabilistic matrix factorization features to train a network based on a generative adversarial network (GAN) [53] with the GRU [54]. More recent work incorporates structural information through graph-based learning. QoSGNN [55] introduces service invocation graph features within an attention-based MF framework to model structural dependencies. Chang et al. [56] constructed user–service graphs using geographical and AS information, partitioned them into subgraphs to capture localized interactions while mitigating noise, and aggregated representations using a Gaussian Mixture Model (GMM) for final prediction. Beyond structural modeling, llmQoS [57] leverages language models to enhance discrete textual features associated with services, further enriching representation learning.

Overall, DA-based approaches progressively extend from basic MLP architectures to hybrid MF–deep models and graph-enhanced frameworks. While they provide stronger representation capability and modeling flexibility, they typically require larger training datasets, careful architectural tuning, and increased computational cost compared to traditional MF and FM methods.

Limitation of Model-based Approaches. However, despite modeling temporal dependencies explicitly, TF-based methods often suffer from a limited ability to capture complex implicit feature interactions, which may lead to reduced prediction accuracy under highly sparse conditions. Although MF and its extensions improve scalability and alleviate data sparsity through latent representation learning, many approaches depend on side information, require careful hyperparameter tuning, and struggle to model complex non-linear relationships inherent in QoS data.

2.2.1.7 Hybrid Approach for QoS Prediction

Hybrid CF approaches combine memory-based and model-based techniques to enhance QoS prediction accuracy. While Memory-based methods exploit user and service similarities, model-based approaches (e.g., MF, FM, and DA) learn latent representations to mitigate sparsity and limited generalization in neighborhood-based models. To integrate these complementary strengths, Chowdhury et al. [26] proposed a context-aware hybrid framework that combines memory-based and model-based predictions through a hierarchical fusion strategy. OffDQ [58] incorporates QoS-driven similarity features into auto-encoder-based network, while NCRL [59] integrates contextual information and historical similarity using a two-tower residual architecture.

Although hybrid methods often achieve improved predictive performance, they still struggle to address multiple anomalies, such as grey-sheep behavior and outliers, within a unified modeling framework.

2.2.2 Challenges in QoS Prediction

Fig. 2.3 illustrates the key challenges faced by CF-based QoS prediction methods in web service environments. Although these approaches demonstrate strong performance in controlled experimental settings, real-world deployment exposes several inherent limitations. These issues stem not only from data sparsity but also from anomalies in QoS invocation logs, limitations in feature representation, scalability constraints, latency requirements, and the complexity of multi-attribute modeling.

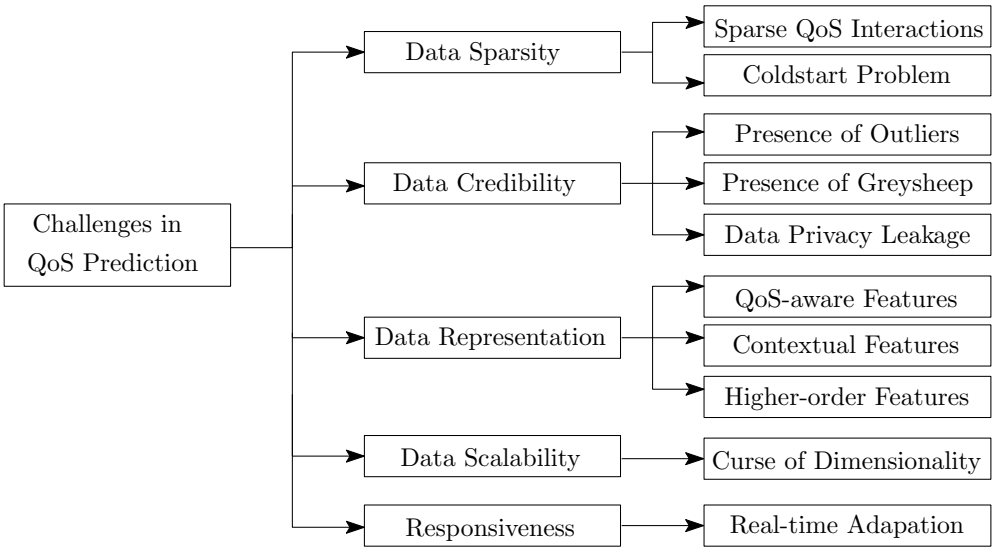


Figure 2.3: Overview of challenges in QoS prediction approaches.

The following subsections discuss these challenges in detail and review representative state-of-the-art methods proposed to address them.

2.2.2.1 Data Sparsity: Challenges and Mitigation Strategies

Data sparsity remains one of the most persistent and fundamental obstacles in QoS prediction. In large-scale service-oriented environments, the number of users and services continues to grow, yet the interaction density between them remains extremely low. A typical user invokes only a limited number of services, and many services are accessed by only a small fraction of users. Consequently, the resulting user–service matrix is overwhelmingly sparse, containing very few observed QoS records relative to the total possible interactions. This lack of sufficient collaborative information severely restricts the ability of learning algorithms to infer reliable patterns. In such conditions, similarity measures become unstable, latent representations become under-trained, and prediction accuracy deteriorates significantly.

An additional and equally critical aspect of sparsity arises from the *cold-start* problem. When new users or newly deployed services enter the ecosystem, historical QoS observations are either minimal or entirely absent. Since most collaborative models rely on past interaction data, cold-start instances often lead to unreliable or

biased predictions. The literature has proposed several strategies to alleviate these challenges, which are discussed below.

2.2.2.1.1 Conventional Approaches. Early memory-based collaborative filtering (CF) methods addressed data sparsity by identifying similar users or services using correlation metrics such as Pearson Correlation Coefficient (PCC) or cosine similarity measure (CSM). Pioneer works, including UPCC [6], IPCC [7], and WSRec [8], rely on neighborhood selection followed by weighted aggregation for prediction. Similarly, time-aware extensions [18, 60–63] incorporate temporal information while retaining similarity-based formulations.

Despite their simplicity and interpretability, these approaches face significant challenges under sparse QoS observations. Limited interactions between users and services hinder reliable similarity estimation, resulting in unstable neighbor selection and degraded prediction accuracy. Moreover, memory-based methods exhibit scalability limitations in large-scale service environments. These constraints motivate the development of more advanced models capable of implicitly alleviating sparsity through latent representation learning and structured modeling.

2.2.2.1.2 Matrix/Tensor Factorization Methods. Matrix Factorization (MF) and Tensor Factorization (TF) methods represent a significant advancement in addressing sparsity in QoS prediction. These approaches uncover latent structures by decomposing the user–service interaction matrix (or tensor) into lower-dimensional representations, enabling prediction even when explicit observations are limited. Nonnegative Matrix Factorization (NMF) [32] and Probabilistic Matrix Factorization (PMF) [35], originally proposed in other domains, have been widely adopted for handling sparsity in QoS data. For dynamic QoS scenarios, TF-based models extend this framework by incorporating time as an additional dimension [42], thereby modeling temporal variations in service performance. To improve generalization and reduce overfitting, several variants introduce regularization mechanisms [17, 42]. Furthermore, context-aware MF models [64–66] integrate auxiliary attributes—such as user

ID, geographical region, and autonomous system (AS), into the factorization process to enhance representation quality.

Although factorization methods alleviate sparsity by learning compact latent representations, they typically assume linear interactions between users and services and may struggle to capture complex higher-order dependencies. Moreover, when interaction data is extremely sparse, the learned latent factors can become unreliable, limiting predictive performance.

2.2.2.1.3 Graph Neural Network (GNN)-based Methods. Graph-based approaches have recently gained attention for addressing sparsity by explicitly modeling relational dependencies among users and services. Typically, graph convolutional networks (GCNs) [67] and graph attention networks (GATs) [68] are applied to bipartite graphs, where users and services are represented as nodes and observed QoS interactions form the edges. GMF [56] introduced a graph-enhanced matrix factorization framework. RIGCN [69] and QoSGNN [55] employed GCNs with attention mechanisms and graph transformers for static QoS prediction. GMCL [70] and BGCL [71] adopted graph contrastive learning to enhance representation robustness. For time-aware modeling, DGNCL [72] integrated GCN with gated recurrent units (GRU) to capture temporal dynamics in QoS interactions. These architectures propagate information through neighborhood aggregation, enabling nodes with limited observations to benefit from higher-order connectivity. As a result, graph-based models alleviate sparsity limitations inherent in traditional MF-based frameworks.

Despite their advantages, deep GNN architectures may suffer from over-smoothing or over-squashing when multiple layers are stacked [73]. Moreover, effective graph construction depends on the availability of reliable interaction data, and extremely sparse graph regions remain difficult to model accurately.

2.2.2.1.4 Clustering-Based Techniques. Clustering-based techniques are employed to mitigate data sparsity by leveraging shared patterns within grouped users or services. In the literature, clusters are typically formed based on similarities in QoS

observations or contextual attributes. QoS prediction is then performed within each cluster, which stabilizes similarity computation, reduces effective dimensionality, and alleviates sparsity [15, 16, 31, 74, 75].

However, the effectiveness of clustering-based methods strongly depends on cluster quality, which is itself influenced by the initial sparsity of the QoS data. Inaccurate cluster assignments may propagate prediction errors, and clusters with few members may still suffer from limited information, reducing their robustness.

2.2.2.1.5 Cold-start: Challenges and Solutions. New users or services are frequently introduced into web service ecosystems, giving rise to the cold-start problem. Since no prior interaction data is available for these entities, specialized modeling strategies are required. Matrix factorization (MF)-based approaches [42, 76] and graph-based methods have been employed to alleviate the impact of cold-start. Several studies leverage contextual metadata—such as geographical information (e.g., region), network attributes, and service provider details (e.g., autonomous system, AS)—to initialize latent representations for cold-start users or services [12, 14, 66, 69, 77–83]. BGCL [71] adopted multiple strategies to address sparsity and cold-start by employing edge-dropout-based data augmentation and incorporating both location similarity and invocation similarity within a graph contrastive learning framework. HTG [84] utilized heterogeneous topology and contextual attributes to enhance representation learning for cold-start entities.

Although contextual augmentation can partially mitigate the cold-start issue, such metadata may not always be available due to privacy constraints [11,23]. Furthermore, contextual similarity does not necessarily translate into performance similarity under dynamic network conditions.

2.2.2.1.6 Other Sparsity-Handling Approaches. Additional strategies include data imputation techniques [26, 46, 50, 58, 85] and trust-aware filtering mechanisms [15, 16]. Some methods pre-fill missing QoS entries prior to model training, while others selectively utilize reliability-aware subsets of data instead of the entire

available interaction set across related tasks. Although these approaches yield incremental improvements, balancing model complexity, scalability, and robustness remains an open challenge.

2.2.2.2 Data Credibility: Issues, Solutions, and Limitations

Beyond sparsity, earlier QoS prediction methods assume that the observed QoS values contributed by users are accurate and reliable. However, in practical deployments, QoS measurements are often affected by system failures, network instability, and measurement errors. In addition, the presence of atypical users or services, often called grey-sheep users or services (GSU/GSS) has unique patterns of invocations [86]. Without addressing such anomalies, it leads to inaccurate QoS prediction. Further, these factors challenge the reliability assumption of conventional QoS prediction models. Such an assumption becomes particularly critical in safety- sensitive applications, including smart transportation systems, autonomous vehicles, and Internet-of-Things (IoT) environments, where inaccurate QoS estimation may directly affect system performance and operational safety. In the following subsections, we discuss the reasons of the reliability failure in the QoS prediction methods broadly into two broad classes: outliers, and grey-sheep.

2.2.2.2.1 Understanding Anomalies in QoS Data.

Definition 14 (Outliers)**.** *Outliers in QoS prediction refer to observations whose values deviate significantly from the expected distribution of user–service interactions. These deviations may arise from transient network congestion, service malfunction, measurement errors, or malicious manipulation. Such values, if untreated, can bias model training and degrade prediction performance.*

Definition 15 (Grey-sheep)**.** *Grey-sheep users or services denote entities whose QoS invocation patterns substantially differ from the majority. Unlike outliers, which are isolated abnormal points, grey-sheep exhibit consistent yet atypical behavioral distributions, making them difficult to model using conventional collaborative assumptions.*

2.2.2.2.2 Outlier Detection and Handling. Many QoS prediction models overlook the presence of outliers in invocation data, leading to degraded predictive accuracy and limited robustness. Since QoS logs often contain abnormal or noisy measurements, robust learning strategies have been introduced to mitigate their impact. A common approach is the adoption of outlier-insensitive loss functions during model training. Robust losses such as L1, Huber, and Cauchy reduce sensitivity to extreme residuals by down-weighting large deviations, thereby preventing anomalous samples from dominating parameter updates.

The **L1 loss** has been widely adopted across different QoS prediction settings, including static methods [59, 87], time-aware models [14, 79, 88, 89], privacy-preserving frameworks [64, 90, 91], and multi-QoS optimization approaches [51, 92]. QoSGNN [55] incorporated L1 loss within a graph neural framework, while QoS-BERT [93] combined Bayesian regression with L1 loss using BERT-based contextual embeddings. DHGCL [94] further integrated L1, contrastive, and KL divergence losses within a graph contrastive learning framework enhanced by spectral clustering and attention mechanisms. PDS-Net [95] adopted Gaussian distribution modeling with KL divergence and L1 regularization to improve robustness.

Huber loss has also been employed to balance sensitivity to prediction errors by treating small and large deviations differently. Small errors are penalized quadratically, whereas large errors are penalized linearly, thereby limiting the influence of extreme outliers. RTF [43] incorporated Huber loss within a time-aware TF framework. It has also been applied in several static QoS prediction models [13, 52, 96]. In addition, llmQoS [97] employed Huber loss in a feed-forward network that leverages large language model (LLM)-based descriptive features to encode user and service contextual attributes.

Furthermore, **Cauchy loss** has been incorporated into robust factorization frameworks due to its logarithmic penalization of large residuals, which further suppresses the influence of extreme outliers. RNL [98] employed Cauchy loss within a non-negative tensor factorization model optimized using the alternating direction method of multipliers (ADMM), achieving improved robustness against severe outliers.

Beyond robust loss design, **regularization techniques** have been incorporated to constrain model complexity and stability, thereby indirectly reducing sensitivity to outliers. Specifically, L2 regularization is widely adopted in dynamic QoS prediction methods [17, 76, 99–105]. INAL [106] introduced an instance-frequency-weighted regularization within a non-negative TF model optimized using particle swarm optimization (PSO). NLFT [107] investigated multiple regularization schemes, including L1, L2, elastic net, logarithmic penalties, and dropout, within non-negative factorization model.

Recent work has combined robust learning with explicit outlier detection mechanisms. OffDQ [58] proposed an iterative detection mechanism based on distributional skewness, identifying extreme values outside statistically defined bounds. A similar strategy was adopted in TRQP [108], where detected outliers were removed during performance evaluation. More comprehensive frameworks integrate robust losses during training and explicit detection during inference. RIGCN [69] employed outlier and pattern measurement techniques alongside L1 loss and regularization within a graph convolutional model. The Isolation Forest (i-Forest) algorithm [109] has been widely used for explicit anomaly detection in QoS datasets. CMF [42] combined Cauchy loss with regularization during matrix and tensor factorization training, followed by i-Forest-based detection. Similar strategies were adopted in HyLoReF [110] and RLMF [65]. HRMI [78] incorporated L1 and mutual information loss during training and applied i-Forest for outlier detection. FSNet [111] also employed Isolation Forest for detection and utilized weighted Huber loss within a feed-forward network.

Existing approaches address outliers through robust loss functions, regularization techniques, explicit anomaly detection, or combinations of these strategies. Although such methods improve robustness, developing a unified framework capable of handling diverse anomaly patterns, including atypical users and services (e.g., grey-sheep behavior), remains an open challenge.

2.2.2.2.3 Grey-sheep Identification and Mitigation. Grey-sheep users or services (GSU or GSS) exhibit QoS invocation patterns that significantly deviate from

the majority. Since collaborative filtering models rely on similarity assumptions across user and services, the presence of such atypical entities can degrade prediction reliability. Addressing this issue requires both accurate identification and appropriate handling mechanisms. Existing studies regarding the grey-sheep could be clubbed into two categories based upon the detection of grey-sheep that avoid improving the prediction model for such instances, and other class of category explicitly mitigate these challenges and introduce mechanism to improve the prediction for such atypical users, thereby improving the overall reliability and prediction performance.

Recent studies have introduced clustering-based reputation models and trust-aware mechanisms to identify entities that exhibit significant deviations from collective QoS patterns. Most of these approaches mitigate their influence by either excluding such entities from similarity computation or incorporating reliability-based weighting schemes during prediction. CAP [16] and TAP [15] employed K-means clustering combined with reputation estimation, while RAP [22] introduced reputation ranking to identify unreliable users. Although effective, ranking-based methods incur increasing computational overhead with large datasets. To integrate reliability into prediction models, RMF [112] incorporated user reputation into the MF-based model. LRMF [77] extended this approach by including location information, though it did not fully preserve the probabilistic nature of reputation scores. DCALF [113] improved detection by applying density-peak clustering over latent factors derived from sparse QoS logs. Trust-aware mechanisms further refined reliability estimation. S-RAP [114] assessed reliability using the 3σ rule and Beta-distribution modeling. TaTruSR [20] introduced local and global trust measures in time-aware neighborhood models. RIGCN [69] proposed deviation-based and KL-divergence-based metrics to quantify reputation iteratively. Recent probabilistic approaches estimate both user and service reliability. RLMF [65] utilized Dirichlet distributions for user reputation but did not model service-side instability. HyLoReF [110] computed joint user–service reputation using utility maximization theory, and HRMI [78] extended this by clustering users and services and generating hybrid probabilistic reputation scores.

After abnormal users or service detection, most approaches address the QoS pre-

diction problem through filtering, down-weighting, or regularization. Specifically, two approaches have been followes: (i) Clustering- and trust-based methods remove unreliable users from similarity aggregation, and (ii) Reputation-aware factorization models (e.g., RMF, LRMF, and HyLoReF), incorporate reliability scores as weights or regularization terms within the optimization objective. RIGCN integrates reputation directly into a robust graph-based loss formulation.

Despite these efforts, most methods rely on exclusion or penalization strategies. Removing grey-sheep entities may discard informative but rare patterns, and dedicated modeling strategies for explicitly learning distinct representations of such entities remain limited.

2.2.2.3 Data Representation: Methods, Challenges, and Feature Design

Effective QoS prediction depends critically on feature representation. Existing methods utilize diverse features, which can be broadly categorized into: (i) Domain-specific features derived directly from QoS interaction logs or contextual attributes of users and services, and (ii) Auto-extracted features learned through representation learning techniques, including spatial, temporal, and spatio-temporal representations.

2.2.2.3.1 Domain-specific Features. Domain-specific features are specialized attributes derived from domain knowledge and application-specific information that capture the unique characteristics of users or services.

Contextual attributes, such as user and service identifiers (IDs), geographical information (e.g., region, latitude–longitude), and network-related attributes (e.g., autonomous system (AS) and IP), are widely adopted in QoS prediction. These attributes are typically encoded as categorical variables using one-hot representations [14, 79, 80]. Although contextual features alone may not achieve performance comparable to interaction-based QoS features, they play a crucial role in mitigating sparsity and cold-start issues. In addition, they complement QoS-derived features by providing auxiliary structural cues.

However, contextual features present limitations. Their availability is not always

guaranteed, particularly in privacy-sensitive environments. Moreover, static contextual similarity does not necessarily imply similarity in service performance under dynamic network conditions.

QoS-derived features are constructed from historical interaction data. Similarity-based measures between users or services are commonly used in hybrid CF frameworks [18, 20, 60–63]. Statistical descriptors, such as mean, variance, and temporal decay weights, are also extracted from historical invocation records. While computationally efficient, these handcrafted features provide limited expressive power and remain sensitive to data sparsity.

To enrich representation capability, latent embedding models—such as matrix factorization [42,76,115] and tensor factorization [17,42,99,100,102,106,116]—have been widely employed. These approaches learn compact representations even under sparse observations. Nevertheless, they primarily model linear interactions and may fail to capture complex nonlinear or structural dependencies.

2.2.2.3.2 Auto-Extracted Features.

To overcome the limited expressiveness of handcrafted features, recent approaches rely on automatic feature learning.

Spatial features capture structural and relational dependencies among entities, such as user–service, user–user, and service–service interactions. Early neural models employed feed-forward networks to generate higher-order nonlinear representations. However, such architectures do not explicitly encode structural or topological relationships. Graph-based models, including GCN [69, 117], GAT [72, 84], and graph contrastive learning frameworks [94], address this limitation by performing neighborhood aggregation, thereby capturing both local and higher-order connectivity patterns in QoS interaction graphs.

Temporal features model the dynamic nature of QoS. Recurrent neural networks (RNNs), including LSTM and GRU, were initially adopted to capture temporal evolution [18,43,79,118]. However, sequential processing may introduce information bottlenecks and gradient instability. Transformer-based architectures [85] mitigate these limitations by modeling long-range dependencies through parallel attention mecha-

nisms, improving scalability and representation capacity.

Spatio-temporal features have recently been incorporated into QoS prediction frameworks to jointly model spatial and temporal representations. By integrating structural relationships with temporal evolution, these features enable time-aware QoS prediction and capture dynamic dependencies among users and services. For example, DGNCL [72] combines graph attention mechanisms with GRU-based temporal modeling, along with ID-based features. Some studies rely primarily on domain-specific attributes [8,20,42], while others combine domain-specific and auto-extracted features for enhanced representation learning [14,18,19,43,79,80,85,118,119].

Despite these advances, designing unified representations that are robust to sparsity, cold-start, dynamic variability, and anomalous behaviors remains a persistent challenge.

2.2.2.4 Scalability: Algorithms, Bottlenecks, and Trade-offs

Scalability is a critical requirement for QoS prediction in modern service-oriented computing environments, where the number of users, services, and interaction records continues to grow rapidly. Real-world platforms, such as cloud systems, and edge infrastructures, involve large-scale and dynamic interaction spaces with highly sparse QoS observations [11,23]. To address these challenges, existing methods have adopted efficient modeling strategies that reduce computational complexity while preserving prediction accuracy.

Model-based approaches, particularly MF and TF techniques, improve scalability by learning compact latent representations instead of computing explicit pairwise similarities [35,42]. Similarly, deep learning and graph-based methods leverage structured representations and neighborhood aggregation to efficiently capture higher-order relationships among users and services [18,55,72]. These approaches significantly enhance scalability compared to traditional similarity-based techniques and enable QoS prediction in increasingly large service ecosystems.

Context-aware and clustering-based approaches further contribute to scalable QoS prediction by incorporating auxiliary information and reducing the effective interaction

space [12, 16, 26].

As service ecosystems continue to expand, scalability remains closely tied to efficient representation learning, model complexity, and training efficiency. The integration of latent factor models, graph-based learning, and structured feature representations has significantly improved the ability of QoS prediction methods to operate at scale. At the same time, efficient optimization strategies, compact representations, and adaptive learning mechanisms play an important role in ensuring that QoS prediction models remain practical and deployable in real-world, large-scale service-oriented environments [14, 70].

2.2.2.5 Responsiveness: Efficiency and Real-Time Considerations

QoS prediction plays a critical role in real-time service selection, recommendation, and orchestration. In time-sensitive applications, QoS prediction must be performed with minimal latency to enable timely decision-making [14, 18]. High prediction latency can delay service selection, degrade system performance, and reduce user satisfaction. Responsiveness primarily depends on inference efficiency, as efficient models must generate QoS estimates quickly to support real-time service environments.

Memory-based CF methods compute similarities between users or services during prediction [6–8]. Although simple and interpretable, these methods suffer from high inference latency, as similarity computation requires scanning large interaction matrices. This makes them unsuitable for large-scale real-time applications [11]. Moreover, similarity computation must be repeated for each query, further reducing responsiveness.

Model-based approaches shift computational complexity from inference to training. MF and deep learning models require longer training time but enable faster inference once trained [18, 35]. Pre-computed latent representations allow efficient prediction through simple vector operations. However, graph-based and transformer architectures improve representation capability at the cost of increased training and inference overhead due to message passing and attention mechanisms [55, 72, 85]. This creates an inherent trade-off between predictive accuracy and responsiveness.

2.2.3 Unified Models for Multi-QoS Prediction

Multi-task learning (MTL), first introduced in [120], is based on the premise that related tasks can benefit from shared representations during joint training. Instead of optimizing each task independently, MTL leverages common structural patterns across tasks to improve generalization and reduce overfitting. In QoS prediction, this paradigm enables the simultaneous estimation of multiple QoS attributes, such as response time, throughput, or reliability, within a unified architecture. Joint modeling is particularly meaningful in service environments where QoS attributes are not independent. Network congestion, server workload, or routing inefficiencies often affect several QoS parameters simultaneously. Modeling them together allows the learning algorithm to capture such inter-attribute correlations, which may otherwise remain unexplored in single-task settings. Despite its advantages, multi-QoS prediction introduces additional challenges. Data sparsity becomes more pronounced when multiple attributes must be estimated jointly. Anomalous observations may impact each attribute differently, complicating robust learning. More critically, QoS attributes typically differ in numerical scale and distribution. When optimized using a shared objective, tasks with larger magnitude gradients can dominate the learning process, resulting in *negative transfer*. This imbalance often leads to unstable training dynamics and degraded performance for certain tasks.

Several recent works have explored MTL-based architectures for joint QoS modeling. DNM [92] models additive and multiplicative cross-context feature interactions, followed by multilayer perceptrons to capture higher-order dependencies. While effective in learning contextual correlations, it employs uniform loss treatment, which does not explicitly address inter-task gradient imbalance. JQSP [117] adopts a graph-based framework with attention mechanisms operating over ID-based features. By leveraging relational structures, it captures user–service dependencies and incorporates an outlier-sensitive loss formulation. However, its reliance on ID embeddings may limit expressiveness in scenarios with rich contextual heterogeneity. MGEN [121] introduces a gated expert network that separates shared and task-specific represen-

tations. This architecture enables flexible feature allocation across QoS attributes. Nevertheless, expert-based designs may suffer from uneven expert utilization, potentially leading to over-specialization or under-utilization. HTG [122] employs graph attention networks (GATs) to integrate topology-aware structural information, particularly targeting cold-start prediction. Although topology-aware modeling improves structural representation, it does not directly regulate task-level gradient imbalance.

To address the task dominance problem, PMT [123] proposes a multi-expert architecture combined with a homoscedastic uncertainty-based Gaussian likelihood formulation. By learning task-specific variance parameters, it adaptively scales sub-task losses during training. Similarly, EEPrNN [124] employs a Gaussian likelihood objective with homoscedastic uncertainty and integrates L2 regularization to stabilize optimization. These probabilistic weighting strategies mitigate scale discrepancies, yet their effectiveness may decrease when the number of QoS attributes increases substantially or when attribute correlations are highly nonlinear. WAMTL [125] introduces dynamic weight averaging (DWA), which adjusts task weights according to historical loss reduction rates within a multi-gate mixture-of-experts framework. While adaptive, DWA relies on short-term loss dynamics, which may fluctuate under sparse or noisy conditions.

Limitation of Multi-QoS Methods. Existing multi-task QoS models still suffer from gradient imbalance due to heterogeneous numerical scales across attributes, which leads to negative transfer despite loss reweighting strategies. Moreover, most frameworks rely on uniform feature sharing in Euclidean spaces and fail to explicitly capture hierarchical or geometric dependencies among QoS attributes. Scalability and robustness under sparse and noisy conditions also remain insufficiently addressed, particularly as the number of QoS dimensions increases.

2.3 Benchmark Datasets

Service-oriented computing research on QoS modeling for various real-world applications has been widely conducted using real-world QoS datasets, providing a mechanism

Table 2.2: Representative joint learning frameworks for joint QoS prediction

Method	Venue	Core Architecture	Feature Modeling	Loss Strategy	Neg. Transfer Mitigation	Structural Modeling
MEQP [126]	Inf. Syst. Frontiers 2014	FM + BiLSTM	Static + Dynamic	Uniform	No	No
DNM [92]	IEEE TSC 2021	Linear Features + MLP	Context interactions	Uniform	No	No
JQSP [117]	IEEE TNSM 2023	Graph + Attention	ID-based	Outlier-sensitive	Partial	Yes (Graph-based)
PMT [123]	IEEE TNSM 2023	Multi-Expert + Attention	Heterogeneous	Homoscedastic uncertainty	Yes	No
MGEN [121]	JPCS 2023	Gated Expert Network	Contextual + QoS	Uniform	No	No
EEPrNN [124]	IEEE Trans. Computers	Attention + FFN	Contextual	Gaussian likelihood + Homoscedastic uncertainty	Yes	No
WAMTL [125]	ICWS 2024	Multi-Gate MoE	ID-based	Dynamic Weight Averaging	Yes	No
HTG [122]	ETT 2024	GAT-based MTL	Topology-aware	Uniform	No	Yes (Topology-aware)

Table 2.3: Statistical summary of benchmark QoS datasets

Attribute	**WSDREAM-1**	**WSDREAM-2**	**Reliability Dataset**
QoS Parameters	RT, TP	RT, TP	RT, TP, RE
No. of Users ($\|\mathcal{U}\|$)	339	142	112
No. of Services ($\|\mathcal{S}\|$)	5825	4500	36
No. of Time Steps ($\|\mathcal{T}\|$)	–	64	–
Total Records	∼1.8M	∼30M	2870
Density	92–94%	60–75%	71.18%
RT Range	0.001 – 19.999	0.001 – 19.999	0.011 – 25.806
TP Range	0.004 – 1000	3.65e-5 – 6726.834	0.648 - 3454.468
RE Range	–	–	0.010–1.000
RT (Mean ± Std)	0.91 ± 1.97	3.18 ± 6.13	1.27 ± 2.52
TP (Mean ± Std)	47.56 ± 110.80	11.34 ± 54.28	35.090 ± 132.533
RE (Mean ± Std)	–	–	0.9897 ± 0.0592
User's Region and AS	31, 137	–, –	24, 80
Service's Region and Providers	74, 2699	–, –	16, 28

for comprehensive evaluation of the frameworks. Zheng et al. [127] investigated personalized QoS prediction and collected three publicly available datasets from the Distributed REliability Assessment Mechanism for Web services (WS-DREAM) project, as described below.

- **Static QoS Modeling Dataset (WSDREAM-1) [1].** This dataset comprises two QoS parameters: Response Time (RT) and Throughput (TP), collected for 339 users and 5825 real-world web services. In addition to QoS measurements, it provides contextual information for both users and services, including geographical regions, autonomous systems (AS), and service provider metadata. This records aggregated QoS observations without explicit temporal indexing and is therefore primarily used for evaluating static QoS prediction frameworks. Although the dataset exhibits relatively high density compared to many real-world production

systems, the large user–service space still results in substantial sparsity, making it suitable for studying collaborative filtering methods and robustness under limited interactions.

- **Reliability Dataset [8].** This dataset extends the evaluation setting to three QoS parameters: RT, TP, and Reliability (RE), collected for the 150 web services invoked by the 100 distributed users. This dataset is often utilized for the evaluation of failure probability and in multi-metric QoS prediction frameworks.

- **Time-Aware Performance Dataset (WSDREAM-2) [127].** QoS values often vary over time due to network instability, fluctuating workloads, and geographical factors. Such temporal dynamics are not captured in previous WSDREAM datasets. To address this limitation, this dataset introduces time-aware QoS modeling by recording QoS values for the same QoS parameters for 142 users interacting with 4,500 real-world web services across 64 consecutive time intervals, with each interval separated by 15 minutes. This temporal structure makes the dataset particularly suitable for evaluating spatio-temporal models and dynamic representation learning approaches. However, this dataset does not contain the contextual details of users and services (e.g., region, AS), thereby does not allow context-aware modeling of QoS.

The statistical characteristics of all datasets are illustrated in Table 2.3.

Limitations of the Datasets. While WSDREAM [1, 8, 127] has played a foundational role in advancing QoS prediction research, its design is primarily aligned with earlier SOAP-based service architectures. In contrast, contemporary service ecosystems increasingly rely on RESTful APIs or gRPC-based communication frameworks, particularly in distributed and edge-oriented environments. As a result, existing benchmarks do not fully reflect the characteristics of modern service deployments. This evolution highlights the need for next-generation QoS datasets that incorporate contemporary service paradigms along with richer contextual attributes to better represent current service computing infrastructures.

2.4 Evaluation Metrics

2.4.1 Error-based Metrics

The performance of QoS prediction models is commonly evaluated using error-based metrics that quantify the deviation between predicted and observed QoS values. Among these, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are the most widely adopted metrics in the literature [14,59,128]. In addition, normalized error measures, relative performance gain, and inference time are often reported to provide a comprehensive assessment of both predictive accuracy and practical applicability.

2.4.1.1 Mean Absolute Error (MAE)

MAE measures the average magnitude of the absolute differences between actual and predicted QoS values and is defined as

$$\mathrm{MAE} = \frac{1}{|\mathrm{TeD}|} \sum_{(u_i,s_j)\in \mathrm{TeD}} |q_{ij} - \hat{q}_{ij}| \,. \tag{2.12}$$

Here, TeD denotes the test dataset, $|\mathrm{TeD}|$ represents the number of test samples, q_{ij} is the observed QoS value, and $\hat{q}_{ij}$ is the predicted value for the user–service pair (u_i, s_j).

MAE assigns equal importance to all prediction errors and therefore provides an intuitive measure of the average deviation between predicted and actual QoS values. However, when QoS attributes differ significantly in scale across datasets, direct comparison of MAE values may be misleading. To address this limitation, MAE can be normalized by the range of observed QoS values, resulting in the Normalized Mean Absolute Error (NMAE), defined as $\mathrm{NMAE} = \mathrm{MAE} \,/\, (\max(q_{ij}) - \min(q_{ij}))$.

By removing scale dependency, NMAE enables fair comparison of prediction performance across heterogeneous QoS attributes and datasets.

2.4.1.2 Root Mean Squared Error (RMSE)

RMSE measures the square root of the average squared prediction error and is given by

$$\text{RMSE} = \sqrt{\frac{1}{|\text{TeD}|} \sum_{(u_i, s_j) \in \text{TeD}} (q_{ij} - \hat{q}_{ij})^2}. \tag{2.13}$$

Unlike MAE, RMSE penalizes larger errors more heavily due to the quadratic term. Consequently, it is particularly informative when large deviations are undesirable, as it emphasizes prediction stability and reliability.

2.4.1.3 Performance Gain

To quantify the comparative advantage of one method over another, the performance gain is employed. Given two methods M_1 and M_2 with performance scores P_1 and P_2 (measured in terms of MAE or RMSE), the improvement of M_1 over M_2 is defined as

$$PG(M_1, M_2) = \left(\frac{P_2 - P_1}{P_2} \right) \times 100\%. \tag{2.14}$$

A positive value of PG indicates that M_1 achieves lower prediction error and thus outperforms M_2, providing a clear quantitative measure of relative gain.

2.4.2 Computational Complexity and Model Efficiency

In addition to predictive accuracy, computational efficiency is an essential consideration in QoS prediction, particularly in large-scale and dynamic service environments. While complex architectures may improve modeling capacity, excessive computational or memory overhead can restrict their practical deployment.

2.4.2.1 Parameter Count

The number of trainable parameters reflects the memory footprint and structural complexity of a model. Although a higher parameter count may increase representational capacity, it also raises computational cost and the potential risk of overfitting. Reporting parameter size therefore enables transparent comparison of model compactness and scalability.

2.4.2.2 Floating-Point Operations (FLOPs)

Floating-point operations (FLOPs) quantify the total number of arithmetic operations required to perform a forward pass during inference. FLOPs provide a hardware-independent estimate of the theoretical computational cost of a model. Lower FLOPs typically indicate greater computational efficiency, which is particularly important in resource-constrained or real-time service environments.

2.4.2.3 Training and Inference Time

Beyond theoretical complexity measures (e.g., FLOPs and parameter count), empirical efficiency is evaluated through training and inference time. Training time represents the total computational cost required to learn model parameters and reflects scalability with respect to dataset size and architectural complexity.

Inference time denotes the average latency required to generate a QoS estimate for a given user–service pair. In service-oriented computing environments, where QoS prediction directly supports real-time service selection, low inference latency is critical to maintaining responsiveness and system usability.

Together, training and inference time provide a practical assessment of model efficiency, capturing both the cost of learning and the feasibility of deployment in dynamic, time-sensitive settings.

Chapter 3

Temporal QoS Prediction Using Multi-Source Collaborative Features

Past QoS prediction methods struggle to capture the complex, high-dimensional features in QoS data sequences. Despite advances in deep learning, these models struggle to capture higher-order triadic relationships among user-service interactions over time due to limitations in feature representation. More recent approaches either used QoS features derived from the temporal QoS invocation log [62, 129] or contextual features (e.g., AS, Region) for QoS prediction, but were unable to conduct a more comprehensive exploration of user-service interactions to enhance prediction accuracy.

This chapter introduces "Temporal QoS Prediction Using Multi-Source Collaborative Features (TPMCF)", a novel framework. In contrast to prior methods, TPMCF distinguishes itself by extensively leveraging collaborative features among users and services. This is achieved by integrating explicit features derived from the QoS invocation log with auto-extracted spatio-temporal features. This dual-feature approach

This chapter is derived from:

- **S. Kumar**, S. Chattopadhyay and C. Adak, "TPMCF: Temporal QoS Prediction Using Multi-Source Collaborative Features," in IEEE Transactions on Network and Service Management, vol. 21, no. 4, pp. 3945-3955, Aug. 2024, doi: 10.1109/TNSM.2024.3395428.

enables TPMCF to capture intricate, higher-order relationships within QoS data, resulting in superior prediction accuracy. Specifically, TPMCF harnesses the power of Graph Convolution [67] in conjunction with a Transformer Encoder [130] to effectively utilize spatio-temporal collaborative features, enabling it to capture the triadic relationships embedded within temporal QoS invocation sequences. TPMCF offers a dual advantage: (a) it excels at capturing multi-source collaborative features, enhancing the accuracy of temporal QoS prediction, (b) it leverages a transformer encoder with multi-head attention to effectively capture temporal dependencies among QoS data. Furthermore, TPMCF, which adopts the Cauchy loss as the objective function, ensures its robustness to outliers. Its offline training of graph convolution and the temporal QoS prediction module has no impact on prediction time, ensuring faster responsiveness. To ensure scalability, the latent feature representation obtained from autoencoders is used to train the proposed graph convolution module in TPMCF.

In summary, the key contributions of this chapter are:

1. **Utilization of multi-source collaborative features for QoS prediction**: TPMCF effectively utilizes collaborative features from both users and services to capture intricate, higher-order relationships among user-service interactions over time, achieving enhanced prediction accuracy. On the one hand, TPMCF combines explicit features obtained from QoS invocation log with auto-extracted features for the target user-service pair. On the other hand, TPMCF auto-extracts spatial features by exploring neighborhood relationships and temporal features by analyzing the user-service interactions over time.

2. **Spatial feature extraction using graph convolution**: We propose a graph convolutional matrix factorization (GCMF) module to auto extract spatial features by investigating the neighborhood of the given user-service pair. On the one hand, GCMF addresses the issue of data sparsity, a prevalent issue in the realm of temporal QoS prediction [11]. On the other hand, GCMF excels at capturing complex relationships within QoS data, thereby enhancing the accuracy of QoS prediction.

3. **QoS prediction using a transformer encoder**: We introduce a temporal QoS

prediction module featuring a transformer encoder followed by a fully connected neural network. The transformer encoder, distinguished by its self-attention mechanism [130], exhibits superior performance compared to sequential architectures relying on LSTM [131] and GRU [18]. We adopt the transformer encoder architecture [130] and propose an enhanced version, the predictive transformer encoder (PTE), designed to capture temporal dependencies in QoS data. The PTE takes spatial features extracted by GCMFs for a given user-service pair, generating spatio-temporal features by exploiting temporal dependencies among the feature sequences, which is the earliest attempt of its kind. Subsequently, a fully connected neural network utilizes these spatio-temporal features for QoS prediction.

4. **Extensive experiments**: We extensively experimented on two benchmark datasets of WSDREAM-2 [127]. Experimental results indicate that our framework outperformed major SOTA approaches in terms of prediction accuracy.

The rest of the chapter is organized as follows. Section 3.1 then discusses the proposed framework in detail. The experimental results are analyzed in Section 3.2 Finally, Section 3.3 concludes this chapter.

3.1 TPMCF: Proposed Framework

Following the time-aware QoS prediction formulation defined in Definition 5, TPMCF aims to address the limitations of existing temporal QoS prediction methods by investigating the following research questions: (i) how to effectively capture collaborative spatial dependencies among users and services by leveraging their interaction graph structure, (ii) how to model temporal dynamics in QoS observations to learn expressive spatio-temporal representations, and (iii) how to perform accurate QoS prediction under sparse interaction conditions without relying on external data imputation techniques. Existing approaches primarily rely on sequential modeling or similarity-based aggregation, which limits their ability to jointly capture structural and temporal dependencies. To address these challenges, TPMCF introduces a unified framework that

integrates graph convolutional matrix factorization for spatial feature extraction and a predictive transformer encoder for temporal modeling, enabling accurate and robust temporal QoS prediction.

This section discusses our proposed framework TPMCF for temporal QoS prediction. TPMCF has two primary modules: (a) *Collaborative spatial feature extraction module (CSFE)*: responsible for extracting spatial features using graph convolutional matrix factorization (GCMF), and (b) *Temporal QoS prediction module (TQP)*: accountable for generating spatio-temporal features utilizing a variant of transformer encoder, followed by QoS prediction using a fully connected neural network. Given the initial feature vectors of a user-service pair u_i, s_j at a specific time-step, GCMF exploits the neighborhood of u_i and s_j to extract spatial features. These spatial features encompass the self-feature of the user or service, along with the features of their neighboring users and services. The Predictive Transformer Encoder (PTE) then utilizes these spatial features to generate spatio-temporal features, which are subsequently employed by a fully connected neural network for QoS prediction.

Apart from enhancing prediction accuracy through spatial feature extraction, GCMF offers another significant advantage when employed prior to the transformer encoder. GCMF adeptly addresses data-sparsity issues, eliminating the need for additional data imputation methods to predict missing values. This is crucial for achieving high prediction accuracy, especially when using deep neural architectures for estimation. We now explain each module of TPMCF in detail. We begin by describing the QoS invocation graph representation, which is one of the primary components the CSFE module of TPMCF.

3.1.1 Representation of QoS Invocation Graph

We first define the QoS invocation graph (QIG).

Definition 16 (QoS Invocation Graph (QIG))**.** *A QoS invocation graph* $\mathcal{G}^t = (U^t \cup S^t, E^t, \mathcal{UE}^t \cup \mathcal{SE}^t)$ *is a bipartite graph, where the vertices* U^t *and* S^t *represent the set of users* $\mathcal{U}$ *and the set of services* $\mathcal{S}$*, respectively. An edge* $e_{ij}^t = (u_i^t, s_j^t) \in E^t$ *exists*

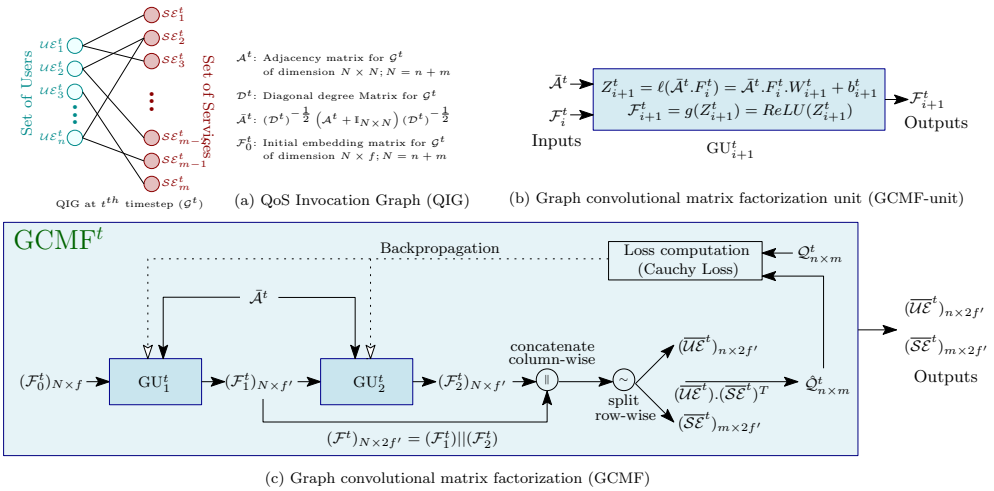


Figure 3.1: Collaborative Spatial Feature Extraction module (CSFE)

between two vertices, $u_i^t \in U^t$ and $s_j^t \in S^t$, if $\mathcal{Q}(i,j,t) = q_{ij}^t \neq 0$. Each node $u_i^t \in U^t$ and $s_j^t \in S^t$ are associated with node vectors $\mathcal{UE}_i^t \in \mathcal{UE}^t$ and $\mathcal{SE}_j^t \in \mathcal{SE}^t$ representing the initial feature embedding of u_i and s_j at time-step t, respectively. ■

An example of a QIG at time-step t is shown in Fig. 3.1(a). In CSFE module, a QIG $\mathcal{G}^t$ is represented by an adjacency matrix $\mathcal{A}^t$ of dimension $N \times N$, where $N = n + m$, as defined below.

$$\mathcal{A}^t = (a_{ij}^t) \in \{0,1\}^{N\times N}; a_{ij}^t = \begin{cases} 1, & \text{if } e_{ij}^t \in E^t \text{ of QIG } \mathcal{G}^t \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

The adjacency matrix $\mathcal{A}^t$ helps to explore the neighborhood of each node to generate the spatial feature embedding for all users and services. However, instead of using $\mathcal{A}^t$ directly in the CSFE module, we normalize $\mathcal{A}^t$ (denoted by $\bar{\mathcal{A}}^t$) as:

$$\bar{\mathcal{A}}^t = (\mathcal{D}^t)^{-\frac{1}{2}}(\mathcal{A}^t + \mathcal{I}_{N\times N})(\mathcal{D}^t)^{-\frac{1}{2}} \tag{3.2}$$

where, $\mathcal{D}^t$ is a diagonal matrix representing the degree of each node of $\mathcal{G}^t$ by its diagonal elements, as defined below.

$$\mathcal{D}^t = (d_{ij}^t) \in \mathbb{Z}_+^{N\times N}; \quad d_{ij}^t = \begin{cases} 1 + \sum_{k=1}^{N} \mathcal{A}^t(i,k), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Normalization is required here to avoid the more significant influence of the higher-degree nodes during learning [67]. This is because the higher degree nodes do not necessarily mean they contribute more to the prediction. It only means they have more interactions at that time-step. Moreover, normalization helps in scaling. In Eq. 3.2, an identity matrix (i.e., $\mathcal{I}_{N\times N}$) is added to take into account the self-influence of a node while extracting the spatial features for it. It may be noted, each non-zero element of $\bar{\mathcal{A}}^t$, i.e., $\bar{\mathcal{A}}^t(i,j)$, is normalized by the square root of the number of invocations of the corresponding user u_i and service s_j at time-step t as recorded in $\mathcal{Q}$, refers to Eq. 3.4.

$$\bar{\mathcal{A}}^t(i,j) = \mathcal{A}^t(i,j) / \left(\sqrt{d^t_{ii}}.\sqrt{d^t_{jj}}\right) \tag{3.4}$$

$\bar{\mathcal{A}}^t$ is finally used for the convolution operation in the CSFE module, which is discussed later in this section.

As mentioned in Definition 16, each node in $(U^t \cup S^t)$ is associated with a node vector representing the feature embedding of the corresponding user or service. We now discuss the construction of the initial feature embedding for the users and the services below.

3.1.1.1 Construction of Initial Feature Embedding

The initial feature embedding of a user or service comprises three distinct types of feature vectors, derived from $\mathcal{Q}$, as elaborated below.

(i) Statistical Features ($\mathcal{F}^t_{St}$): For each user $u_i \in \mathcal{U}$ or service $s_j \in \mathcal{S}$, we obtain the statistical features $\mathcal{F}^t_{St}(u_i)$ or $\mathcal{F}^t_{St}(s_j)$ consisting of 5 statistical parameters: minimum, maximum, median, mean, and standard deviation from their QoS invocation profile $\mathcal{Q}^t(u_i)$ or $\mathcal{Q}^t(s_j)$, where $\mathcal{Q}^t(u_i) = \mathcal{Q}(i,.\,,t)$ and $\mathcal{Q}^t(s_j) = \mathcal{Q}(.\,,j,t)$ represent the QoS invocation vectors of u_i and s_j at time-step t, respectively. Statistical features of a user or service capture the global characteristics of the invocation pattern of that user or service.

(ii) QoS Features ($\mathcal{F}^t_Q$): We use matrix decomposition [32] of $\mathcal{Q}^t$ (i.e., QoS invocation log matrix at t) to obtain the QoS feature embedding of length f_q for each user and

service (say, $\mathcal{F}_Q^t(u_i)$ and $\mathcal{F}_Q^t(s_j)$) at t. While statistical features effectively capture the overall characteristics of users and services, they may lack the granularity found in raw QoS invocation data. The latter can be high-dimensional, so we use latent features obtained through non-negative matrix factorization. An additional advantage is the applicability of these features to new users or services, addressing the cold-start problem where there is limited data for statistical feature extraction.

(iii) Collaborative features using correlations ($\mathcal{F}_C^t$): For each user $u_i \in \mathcal{U}$ or service $s_i \in \mathcal{S}$, we first obtain the correlation between u_i or s_i and every other user $u_j \in \mathcal{U}$ or service $s_j \in \mathcal{S}$ in terms of their QoS invocation profile $\mathcal{Q}^t(u_i)$ or $\mathcal{Q}^t(s_i)$ using cosine similarity [132]. It may be noted that for each u_i or s_i, we have a vector of length n or m, which is then fed to a stacked autoencoder [133] to obtain the latent representation of correlation feature embedding of length f_c for each user and service (say, $\mathcal{F}_C^t(u_i)$ or $\mathcal{F}_C^t(s_j)$) at t. This effectively captures collaborative similarity between users or services by utilizing correlation-based features. It leverages the latent representation obtained through the stacked autoencoder to enhance the accuracy of predictions, contributing to a more comprehensive understanding of collaborative relationships in the QoS prediction model.

The final feature embedding for a user u_i or service s_j is constructed by concatenating the above three feature vectors, as shown in Eqs. 3.5-3.6.

$$\mathcal{UE}_i^t = \mathcal{F}_{St}^t(u_i)||\mathcal{F}_Q^t(u_i)||\mathcal{F}_C^t(u_i) \tag{3.5}$$

$$\mathcal{SE}_j^t = \mathcal{F}_{St}^t(s_j)||\mathcal{F}_Q^t(s_j)||\mathcal{F}_C^t(s_j) \tag{3.6}$$

The initial feature embedding matrix $\mathcal{F}_0^t$ has dimensions $N \times f$, with $f = 5 + f_q + f_c$. The first n rows correspond to user-feature embeddings for n users, while the last m rows represent service-feature embeddings for m services. The CSFE module employs this embedding matrix $\mathcal{F}_0^t$ to incorporate neighborhood information for each user and service.

3.1.2 Collaborative Spatial Feature Extraction (CSFE) Module

We now discuss the automated feature generation using GCMF at time-step t. GCMF is adept at operating on sparse user-service interaction matrices, effectively mitigating sparsity in data. Its approach involves capitalizing on localized connectivity patterns and aggregating information from neighboring nodes. By accounting for local structures and indirect connections through shared neighbors, GCMF excels in capturing meaningful relationships, even within sparse regions of the graph. Additionally, parameter sharing, embedding latent features, and employing regularization techniques further enhance GCMF's ability to handle sparsity, facilitating robust generalization and accurate predictions, especially in scenarios with limited direct connections. Fig. 3.1(b) shows the details of a GCMF-unit (GU), which is the core component of the CSFE module. Given the normalized adjacency matrix $\bar{\mathcal{A}}^t$ and feature embedding matrix $\mathcal{F}_i^t$ to the i^{th} GCMF-unit (GU_i^t), for each node $v_j^t \in U^t \cup S^t$ of $\mathcal{G}^t$, GU_i^t combines the feature of v_j^t with the features of all the nodes reachable from v_j with a path length less than or equal to i to obtain spatial feature vector for v_j^t.

In our CSFE module, two GUs are connected sequentially. To avoid the over-smoothing problem [134], the output of GU_1^t (i.e., $\mathcal{F}_1^t$) is concatenated with the output of GU_2^t (i.e., $\mathcal{F}_2^t$) to obtain a matrix $\mathcal{F}^t$ of size $N \times 2f'$. Finally, $\mathcal{F}^t$ is split row-wise to generate user and service embedding matrices, $\overline{\mathcal{UE}}^t$ and $\overline{\mathcal{SE}}^t$, respectively. Fig. 3.1(c) presents the overview of the GCMF architecture. The output of the CSFE module is forwarded to the subsequent module of our framework.

3.1.3 Temporal QoS Prediction (TQP) Module

This is the final module of our framework. Given a target user-service pair u_i and s_j, the objective of this module is to predict the QoS value for u_i, s_j at time-step t. The main crux of this module is to predict the value of $\mathcal{Q}(i, j, k)$ while considering the temporal dependencies for the last $\mathcal{T}$ time-steps.

The first step of this module is to generate the input embedding, which is then

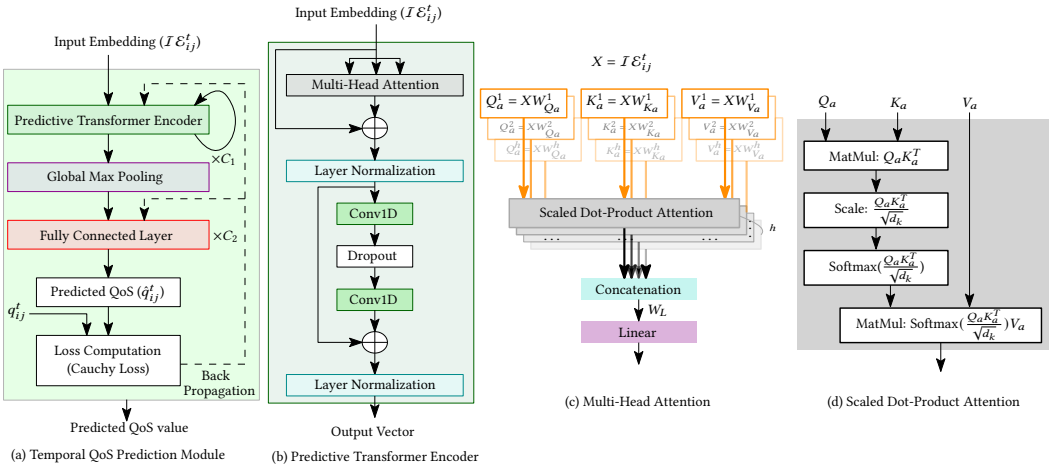


Figure 3.2: Temporal QoS Prediction Module (TQP)

fed to a series of transformer encoder blocks, followed by global max-pooling and fully connected (FC) layers to predict the value of $\mathcal{Q}(i, j, k)$. The pictorial overview of this module is shown in Fig. 3.2(a). We now discuss each step in detail.

3.1.3.1 Input Embedding Construction

At first, for each time-step k (where, $t - \mathcal{T} + 1 \leqslant k \leqslant t$), the user embedding $\overline{\mathcal{UE}}_i^k \in \overline{\mathcal{UE}}^k$ and the service embedding $\overline{\mathcal{SE}}_j^k \in \overline{\mathcal{SE}}^k$ obtained from the CSFE module are concatenated to produce a vector $\overline{\mathcal{E}}_{ij}^k$ of length $4f'$. Finally, $\overline{\mathcal{E}}_{ij}^k$ across all $\mathcal{T}$ time-steps are concatenated row-wise to generate the input embedding $\mathcal{IE}_{ij}^t$ of dimension $\mathcal{T} \times 4f'$.

$$\mathcal{IE}_{ij}^t = \underbrace{((\overline{\mathcal{UE}}_i^{t-\mathcal{T}+1} || \overline{\mathcal{SE}}_j^{t-\mathcal{T}+1})}_{\overline{\mathcal{E}}_{ij}^{t-\mathcal{T}+1}} ||_R \ldots ||_R \underbrace{(\overline{\mathcal{UE}}_i^{t-1} || \overline{\mathcal{SE}}_j^{t-1})}_{\overline{\mathcal{E}}_{ij}^{t-1}} ||_R \underbrace{(\overline{\mathcal{UE}}_i^{t} || \overline{\mathcal{SE}}_j^{t}))}_{\overline{\mathcal{E}}_{ij}^{t}} \tag{3.7}$$

3.1.3.2 Training of PTE

We train the PTE with the input embedding $\mathcal{IE}_{ij}^t$ and target value q_{ij}^t, for all u_i, s_j such that $\mathcal{Q}(i, j, t) \neq 0$. Fig. 3.2(b) presents the detail of our transformer encoder network. The core component of PTE is multi-head attention (MHA) with h heads as presented in Fig. 3.2(c). Each head of MHA comprises scaled dot-product attention (SDPA) [130], which is presented in Fig. 3.2(d). The MHA block's objective is to compute attention across different time-steps for various parts of the input. Each head,

indexed by $i \in 1, 2, \ldots, N_h$, independently calculates a tuple (Q_a^i, K_a^i, V_a^i) comprising query, key, and value, as shown below.

$$\begin{aligned}(Q_a^i)_{\mathcal{T}\times d_k} &= (\mathcal{IE}_{ij}^t)_{\mathcal{T}\times 4f'}\ (W_{Q_a}^i)_{4f'\times d_k},\\ (K_a^i)_{\mathcal{T}\times d_k} &= (\mathcal{IE}_{ij}^t)_{\mathcal{T}\times 4f'}\ (W_{K_a}^i)_{4f'\times d_k},\\ (V_a^i)_{\mathcal{T}\times d_v} &= (\mathcal{IE}_{ij}^t)_{\mathcal{T}\times 4f'}\ (W_{V_a}^i)_{4f'\times d_v}\end{aligned} \tag{3.8}$$

The tuple (Q_a^i, K_a^i, V_a^i) is then forwarded to the SDPA block, which is responsible for computing the attention for the input sequence across $\mathcal{T}$ time-steps. The outcomes of each SDPA block of size $\mathcal{T} \times d_v$ are now concatenated column-wise across all heads, and fed to the next block to perform a linear operation with weight matrix W_L. It may be noted, the MHA block produces an output of size $\mathcal{T} \times 4f'$.

The PTE incorporates a residual connection, adding the output from the MHA with $\mathcal{IE}_{ij}^t$. This combined output undergoes layer normalization (LN) [130]. The normalized output is then processed through two sequentially connected 1D convolution layers (Conv1D), with a dropout layer for regularization [135] in between. The output of the second Conv1D is added to the output of the previous LN, and the sum is passed through another LN to generate the final output vector of the PTE.

In TQP, the input embedding passes through C_1 PTE blocks sequentially. The output from the final PTE block is then sent to a global max pooling layer, followed by C_2 fully connected layers for predicting the value $\hat{q}_{ij}^t$. Moreover, the detailed configuration of TQP is referred to in Section 3.2. It is important to note that the training of TPMCF is conducted in an offline mode, ensuring that training time does not affect its runtime performance. The model is not retrained with each new QoS data incorporation but is updated after a defined interval, such as when matrix density increases by a certain percentage. This efficient strategy balances computational resources and allows the model to adapt to evolving patterns in QoS data.

3.1.4 Outlier Detection and Handling

The QoS prediction accuracy is highly sensitive to the outliers present in the dataset [42]. We employ two different strategies to handle the outliers. To train GCMF

and PTE, we use Cauchy loss [136] to handle outliers, as presented in the following equations.

$$\begin{aligned} \mathcal{L}^k_{GCMF} &= \sum_{q^k_{ij} \neq 0} \log\left(1 + \left((q^k_{ij} - \hat{q}^k_{ij})/\gamma_s\right)^2\right), \\ \mathcal{L}_{PTE} &= \sum_{q^t_{ij} \neq 0} \log\left(1 + \left((q^t_{ij} - \hat{q}^t_{ij})/\gamma_t\right)^2\right) \end{aligned} \tag{3.9}$$

where, γ_s and γ_t are adjustable hyper-parameters. Furthermore, we employ the isolation forest algorithm [109] to detect the outliers present in the dataset for a given outlier ratio λ, where λ is another hyper-parameter used to decide the percentage of outliers required to be removed to evaluate the performance of our framework. In our experiment, we show the performance of TPMCF for different values of λ.

3.2 Experiments

This section presents the experimental setup, model configuration details, and a comprehensive performance evaluation with state-of-the-art (SOTA) methods for dynamic QoS prediction.

The implementation of our framework was performed using TensorFlow v2.6.2 with Python 3.6.9. For training purposes, NVIDIA's Quadro RTX 3000/PCIe/SSE2 GPU with 1920 cores, and 6 GB memory were used. We evaluate the trained model using i9-10885H @ 2.40 GHz×16 processor with x86_64 CPU with 128 GB RAM.

3.2.1 Experimental Setup

We employ WSDREAM-2 [127] dataset, as illustrated in Sec. 2.3. In our experimentation, we used two different train-test split-ups. We divided each dataset into 10% : 90% and 20% : 80% of training-testing ratios for our experiment. We use the notation RT-x (TP-x) to denote the training-testing split as (x% : (100-x)%).

To evaluate our model, we utilize MAE metric defined in Sec. 2.4.1. Each experiment was performed five times. Finally, the average values were recorded and presented in this chapter.

Table 3.1: Configuration of TPMCF

Parameters		Values	
QoS Features and Collaborative features		f_q, f_c	100, 50
GCMF	No. of GConv Units	2	
	Dimension of weight matrices	W_1^t, W_2^t	155 × 64, 64 × 64
	Dimension of automated feature vector	$2f'$	128
PTE	No. of time-steps	$\mathcal{T}$	8
	No. of PTE blocks	C_1	4
	No. of heads	h	4
	Dimension of query, key, and value	d_k, d_k, d_v	256, 256, 256
	First Conv 1D	No. of filters	4
		Filter size	3×3
	Second Conv 1D	No. of filters	$\mathcal{T}$
		Filter size	1×1
	No. of fully connected (FC) layers	C_2	2
Optimizer (GCMF, PTE)		AdamW [137]	

3.2.1.1 Configuration of TPMCF

We now present the details of various hyper-parameters of TPMCF. The configuration details for the GCMF and PTE are presented in Table 3.1. We used λ (outlier ratio) = 0.1 throughout our experiments unless otherwise specified. We also present a few experiments changing the values of some hyper-parameters to show their impact on prediction accuracy.

3.2.2 Performance Analysis

We now analyze the performance of TPMCF and demonstrate the comparative study with respect to the major SOTA methods.

3.2.2.1 Performance of TPMCF and Comparison with SOTA

Here, we present the performance of TPMCF in terms of prediction time and accuracy, as well as the training time of the models. Table 3.2 presents the comparison between SOTA and our framework in terms of prediction accuracy. It is important to note that the results reported in Table 3.2 for various previous methods are obtained after removing 10% or more outliers. In this table, we have categorized the SOTA methods

Table 3.2: Performance of TPMCF and comparison with SoTA in terms of prediction accuracy (MAE)

Feature Category	Methods	Loss	RT-10	RT-20	TP-10	TP-20
QoS	TASR [60]	-	2.8188	2.7120	4.3265	3.6419
	WSPred [17]	MSE	2.4990	2.3000	8.0131	7.6000
	OPST [131]	MSE	1.1722	1.1587	3.2762	3.1193
	BNLFT [99]	MSE	1.0828	1.0575	1.4241	1.3935
	NNCP [116]	MSE	1.0796	1.0536	2.6401	2.5797
	WLRTF [138]	MAE	1.0560	1.0437	2.9576	2.9569
	RNCF [139]	MSE	1.0100	0.9580	-	-
	CTF [42]	Cauchy	0.9215	0.8981	1.3567	1.1945
	DGNCL [72]	MSE	0.9001	0.7419	-	-
	CARP [140]	MSE	0.7709	0.6992	-	-
	PLMF [118]	MSE	0.7580	0.7390	4.1230	4.0320
	TUIPCC [62]	-	0.7578	0.7223	3.7573	3.7177
	DeepTSQP [18]	MSE	0.7539	0.6195	4.8194	4.4519
	RTF [43]	Huber	0.6300	0.5350	4.0250	3.799
QoS+Contextual	Mul-TSFL [80]	-	0.8972	0.8097	-	-
	QSPC [79]	MAE	0.8341	0.8231	3.0604	2.9624
QoS	**TPMCF**	Cauchy	**0.4973**	**0.3864**	**1.0101**	**0.7881**
QoS	*PG*(%)		**21.06%**	**27.96%**	**25.55%**	**34.02%**
QoS+Contextual	*PG*(%)		**40.38%**	**52.28%**	**66.99%**	**73.40%**
nMAE of TPMCF			0.1565	0.1216	0.0890	0.0695

*__Improvement__ over the second-best method

in terms of their usage of features: (a) *Q:* methods that used only QoS features derived from the QoS invocation log $\mathcal{Q}$, and (b) *Q+C:* methods employing the contextual information of users and services along with the QoS features.

3.2.2.1.1 Comparison with SOTA in terms of prediction accuracy. As evident from Table 3.2, TPMCF outperformed all the SOTA methods.

(i) We observed an improvement of 21.06%, 27.96%, 25.55%, and 34.02% of TPMCF over the second best SOTA with QoS features as reported in Table 3.2 on datasets RT-10, RT-20, TP-10, and TP-20, respectively. Similarly, we reported an improvement of our framework over the second-best SOTA with both the QoS and contextual features in Table 3.2.

(ii) We compare TPMCF with several methods utilizing outlier-resilient robust loss

functions, including CTF [42], RTF [43], WLRTF [138], and QSPC [79]. These methods employ Cauchy loss, Huber loss, or MAE (i.e., L1 norm) as robust loss functions. RTF and CTF demonstrate the best prediction accuracy among these methods on the RT and TP datasets, respectively. TPMCF, however, achieved an average improvement of 50.75% and 40.64% across all four datasets compared to RTF and CTF, respectively. This underscores the effectiveness of TPMCF in terms of model performance and the utilization of multi-source collaborative features.

(iii) The final row of Table 3.2 shows the normalized MAE (i.e., nMAE) of TPMCF for all four cases of RT and TP datasets, which also shows that our framework is generalized enough.

3.2.2.1.2 Comparison with SOTA in terms of learning time. Fig. 3.3 shows the comparison between SOTA and our framework in terms of the training time of the models. We have the following observations from Fig. 3.3.

(i) TPMCF had less training time as compared to four SOTA methods (i.e., NNCP [116], BNLFT [99], WLRTF [138], AND CTF [42]). Out of these four methods, NNCP [116] is the fastest in terms of training time. TPMCF achieved 1.12× speed-up and an average improvement of 62.11% over all four datasets as compared to NNCP.

(ii) In comparison to CTF [42], TPMCF exhibits a notable 1.14× speed-up and an average improvement of 40.64% across all four datasets, establishing its efficacy in terms of prediction time and accuracy.

(iii) TPMCF was slower than PLMF [118] and TASR [60] with speed-degradation of 0.6× and 0.2×, respectively. However, on average, TPMCF achieved an improvement of 33.38% and 80.78% over PLMF and TASR, respectively.

3.2.2.1.3 Comparison with SOTA in terms of prediction time. TPMCF undergoes offline training, ensuring that the extended training time does not impact its real-time performance. The performance of TPMCF is gauged through the prediction

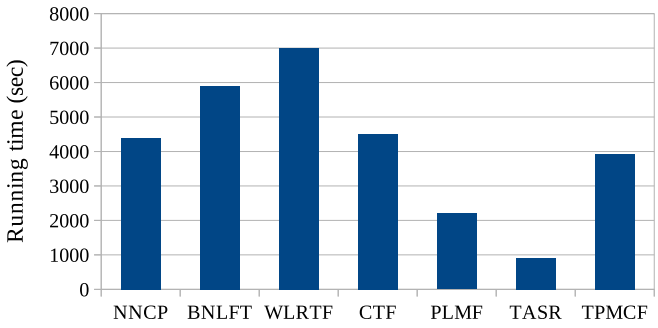


Figure 3.3: Training time comparison of TPMCF with SOTA

Table 3.3: Performance of TPMCF with various loss functions and impact of outliers with Cauchy loss (MAE)

			Outlier ratio (λ)					
			0	0.02	0.04	0.06	0.08	0.1
Segment-1	Loss	MSE	0.9419	0.8443	0.7904	0.7425	0.7063	0.6796
		MAE	0.8430	0.7421	0.6887	0.6417	0.6057	0.5800
		Huber	0.8409	0.7384	0.6869	0.6392	0.6018	0.5742
		Cauchy	0.8132	0.6985	0.6279	0.5670	0.5255	0.4973
Segment-2	Datasets	RT-10	0.8132	0.6985	0.6279	0.5670	0.5255	0.4973
		RT-20	0.6159	0.5326	0.4834	0.4399	0.4072	0.3864
		TP-10	6.4200	2.7457	1.7873	1.3801	1.1490	1.0101
		TP-20	5.4620	2.3853	1.5100	1.1254	0.9179	0.7881

time, with an average of 4.1×10^{-4} seconds—a satisfactory metric when juxtaposed with the service's minimum response time of 10^{-3} seconds.

3.2.2.2 Outlier Analysis

We demonstrate the robustness of TPMCF in handling outliers in Segment-1 of Table 3.3, focusing on the RT-10 dataset. The performance is evaluated with four different loss functions across various outlier ratios (λ). Notably, mean-squared-error (MSE) [141] is sensitive to outliers, resulting in the worst performance when employed as the loss function for TPMCF. In contrast, mean-absolute-error (MAE), Huber loss [142], and Cauchy loss exhibit greater resilience to outliers. Among these, Cauchy loss [136] outperformed the others, as evident in Segment-1 of Table 3.3.

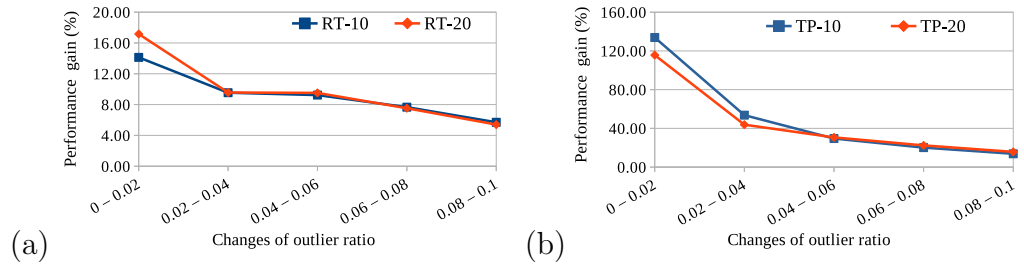


Figure 3.4: Performance gain with change in outlier ratio λ on (a) RT and (b) TP datasets

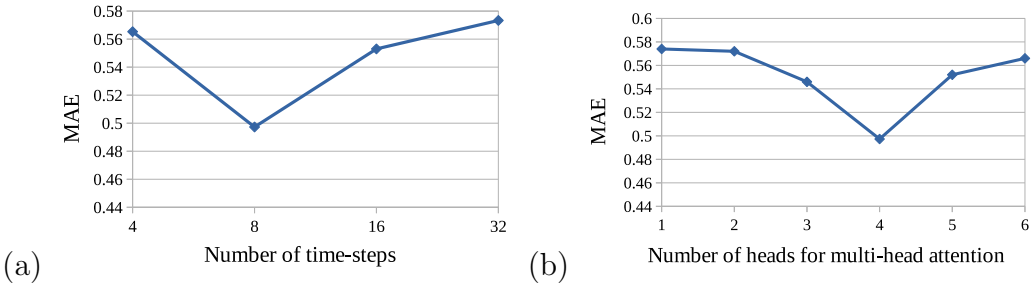


Figure 3.5: Impact of number of (a) time-steps ($\mathcal{T}$), and (b) heads (N_h) on RT-10

Segment-2 of Table 3.3 illustrates the influence of outliers on the prediction accuracy of TPMCF across all four datasets. The results in Segment-2 of Table 3.3 highlight a significant impact of outliers on TPMCF's performance. Fig. 3.4 further visualizes the performance gain (PG) with increasing outlier ratios (λ). Notably, the removal of the initial 2% outliers ($\lambda = 0.02$) yields the maximum performance gain.

As we removed more outliers, the performance gain gradually decreased, as evident from decreasing trends of the curves shown in Figures 3.4(a)-(b). With the change in λ from 0.08 to 0.1, we achieved the least performance gain. In other words, the initial 2% outliers influenced the performance measure (i.e., MAE value) more than the rest. The rate of change of performance improvement decreased with the increase in the outlier ratio. This explains that the outlier detection algorithm is powerful enough to identify the appropriate set of outliers.

Table 3.4: Study on model selection on RT-10 (MAE)

GAT [68]+PTE	GConv [108]+PTE	GCMF+LSTM [143]	GCMF+GRU [54]	TPMCF
1.1161	0.8092	0.6111	0.6171	0.4973

Table 3.5: Module and feature ablation study (MAE)

	Features		RT-10	RT-20	TP-10	TP-20
Module	Spatial	GCMF	0.5884	0.4762	1.1616	0.8877
	Temoral	PTE	0.5842	0.4796	1.1879	0.8506
	Spatial + Temporal	TPMCF	0.4973	0.3864	1.0101	0.7881
Feature	Statistical		0.5939	0.4517	1.4249	1.1112
	QoS		0.5821	0.4093	1.1350	0.8281
	Collaborative (using Correlations)		0.5771	0.4104	1.0891	0.8590
	Combined Features		0.4973	0.3864	1.0101	0.7881

3.2.2.3 Ablation Study

We now present the rationale behind our model selection and the justification for having various modules of TPMCF through the ablation study.

3.2.2.3.1 Model Selection. Table 3.4 presents the performance of TPMCF in terms of MAE as compared to other models that can be used to generate spatial and temporal features. We have the following observations from Table 3.4.

(i) We first compare GCMF with graph attention network (GAT) [68] and graph convolutional network (GConv) [108] on the RT-10 dataset. As evident from Table 3.4, TPMCF achieved an improvement of 55.44% and 38.54% over GAT+PTE and GConv+PTE architectures, respectively. This experiment shows the effectiveness of GCMF in TPMCF.

(ii) We further implemented the temporal QoS prediction block using LSTM [143] and GRU [54]. TPMCF achieved an improvement of 18.62% and 19.41% over GCMF+LSTM and GCMF+GRU architectures, respectively. This experiment explains the importance of the PTE as part of the temporal QoS prediction block.

3.2.2.3.2 Module Ablation. The first three rows of Table 3.5 present the ablation study for various modules in TPMCF. We have the following observations from the first segment of Table 3.5.

(i) TPMCF achieved on average 14.65% improvement over the stand-alone GCMF, which justifies the requirement of temporal QoS prediction block.

(ii) TPMCF achieved on average 14.16% improvement over the stand-alone transformer encoder with temporal features. This implies the necessity of capturing the spatial features along with the temporal dependencies.

3.2.2.3.3 Feature Ablation. The second segment of Table 3.5 presents the feature ablation study for TPMCF. As observed from Table 3.5, TPMCF with the combined features outperformed the same model with individual feature categories, as discussed in Section 3.1.1.1. On the one hand, this experiment shows the significance of using the collaborative features derived from the QoS invocation log apart from the other auto-extracted features used for QoS prediction. On the other hand, it justifies using the different sets of features extracted to employ in TPMCF.

The feature ablation study reveals that the combined explicit features outperformed individual features. The module ablation study provided justification for using spatio-temporal features over individual spatial or temporal features. The model selection experiment elucidated the reasoning behind choosing PTE over traditional LSTM or GRU, as well as the justification for employing GCMF instead of graph attention network (GAT) or graph convolution network (GConv), as used in a method in the literature. In summary, our experimental results showcase that TPMCF excels over major state-of-the-art methods in terms of prediction accuracy while maintaining faster responsiveness. Additionally, we emphasize the importance of each component of TPMCF in contributing to the overall prediction accuracy.

3.2.2.4 Impact of Hyper-parameters

This section assesses how hyperparameters affect TPMCF performance.

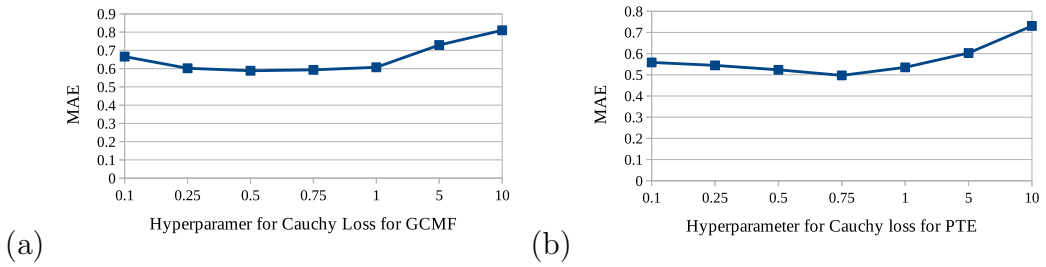


Figure 3.6: Impact of (a) γ_s on performance of GCMF, and (b) γ_t on performance of TPMCF on RT-10 dataset

3.2.2.4.1 Impact of the number of time-steps. Fig. 3.5(a) shows the MAE obtained by TPMCF on the RT-10 dataset with the change in the number of time-steps ($\mathcal{T}$) considered to construct the input embedding for TQP. We achieved the best performance for $\mathcal{T} = 8$. As observed in Fig. 3.5(a), with increasing the value of $\mathcal{T}$, the MAE decreased until $\mathcal{T} = 8$, and then it started increasing for $\mathcal{T} = 16$ and 32.

3.2.2.4.2 Single vs. Multi-Head Attention. To show the impact of the number of heads (N_h), we varied N_h from 1 to 6. The results are reported in Fig. 3.5(b) for the RT-10 dataset. Initially, with the increase in the value of N_h, the MAE value decreased till $N_h = 4$, and it started increasing thereafter, as evident from Fig. 3.5(b).

3.2.2.4.3 Impact of γ_s and γ_t. Fig. 3.6(a) shows the impact of γ_s on the performance of GCMF on the RT-10 dataset. As observed from the figure, GCMF achieved the best prediction accuracy for $\gamma_s = 0.5$. Similarly, Fig. 3.6(b) shows the impact of γ_t on the performance of TPMCF on the RT-10 dataset while keeping γ_s as constant. As observed from Fig. 3.6(b), TPMCF achieved the best prediction accuracy for $\gamma_t = 0.75$.

3.3 Summary

This chapter presents an efficient solution for temporal QoS prediction for service recommendation. Specifically, our framework TPMCF leverages multi-source collaborative features comprising explicit features derived from QoS invocation log and

auto-extracted spatio-temporal features for capturing the triadic relationships among the users, services and time-steps. TPMCF includes a collaborative spatial feature-extraction module (CSFE) and a temporal QoS prediction module (TQP). CSFE is responsible for spatial feature extraction using graph convolutional matrix factorization (GCMF). On the other hand, TQP is accountable for temporal feature extraction using a predictive transformer encoder (PTE), followed by QoS prediction using a fully connected neural network. Additionally, TPMCF is competent in addressing a few fundamental challenges in temporal QoS prediction. For example, to deal with the outliers present in the datasets, TPMCF employs Cauchy loss to train the GCMF and PTE. TPMCF is sparsity-tolerant without using any additional data imputation method due to GCMF. The offline training of GCMF and PTE ensures the high responsiveness of TPMCF. This makes TPMCF a better fit for real-time applications. Moreover, the feature dimensionality reduction with the help of autoencoders makes TPMCF highly scalable. Through extensive experiments on WSDREAM-2 datasets, we have demonstrated that TPMCF attains superior prediction accuracy compared to major state-of-the-art methods. Additionally, TPMCF exhibits reasonably faster training times in comparison to contemporary methods and negligible prediction times relative to the response time of services. The overall findings underscore TPMCF's robustness and efficiency as a solution for temporal QoS prediction, showcasing its potential applicability in the dynamic landscape of web services.

3.4 Limitations

Although TPMCF significantly improves temporal QoS prediction by learning collaborative spatio-temporal representations, it has several important limitations.

(i) TPMCF employs separate architectures for spatial and temporal modeling, where the GCMF module is first trained to extract spatial representations, which are then used as input to the predictive transformer encoder for temporal QoS prediction. Since these components are optimized independently rather than jointly, the framework cannot fully capture the interdependence between structural relationships and

temporal dynamics. Consequently, this sequential design may limit the ability of TPMCF to learn coherent spatio-temporal representations of evolving user–service interactions.

(ii) TPMCF does not explicitly model or identify atypical users or services (referred to as grey-sheep) whose QoS invocation patterns deviate significantly from collaborative trends. The presence of such atypical entities may distort collaborative relationships and negatively affect prediction accuracy.

These limitations motivate the development of a more robust and anomaly-resilient QoS prediction framework capable of jointly modeling collaborative relationships, handling atypical entities, and improving prediction reliability under real-world conditions.

Chapter 4

Anomaly Resilient Real-time QoS Prediction Framework with Graph Convolution

TPMCF introduced a graph-based solution to mitigate sparsity, but it is incapable of handling other anomalies, such as grey-sheep [42, 86], present in QoS data. This chapter presents ARRQP, which not only enhances our predictive performance but also prioritizes the resolution of other anomalies, including outliers, the cold-start problem, and the presence of grey-sheep instances [86]. Specifically, we introduce several innovative solutions to mitigate QoS prediction challenges including: *(i)* We propose a simple yet effective solution leveraging graph convolution [67], which excels in dealing with high sparsity by aggregating and sharing neighborhood information of the nodes, *(ii)* Integrating both contextual information and QoS data, along with spatial collaborative information, offers a holistic perspective on user-service interactions here. This comprehensive approach enhances the system's ability to understand user behaviors and service performance, evidently resulting in more accurate predictions,

---

This chapter is derived from:

- **Suraj Kumar** and Soumi Chattopadhyay, "ARRQP: Anomaly Resilient Real-Time QoS Prediction Framework With Graph Convolution," in IEEE Transactions on Services Computing, vol. 18, no. 3, pp. 1245-1261, May-June 2025, doi: 10.1109/TSC.2025.3565376.

(iii) The outlier-resilient Cauchy loss enables us to minimize the impact of outliers during the model training, *(iv)* The grey-sheep detection block in ARRQP empirically shows the effectiveness in identifying grey-sheep users/services. Importantly, it has the advantage of being able to handle the issue of data sparsity because it does not rely on measuring similarity between users/services., *(v)* In addition to the above, ARRQP offers an effective model specifically designed for predicting the QoS values of grey-sheep users or services, and *(vi)* The CRRQP block is specifically designed to address the prediction of QoS for newly added users/services. It recognizes the challenges posed by the cold-start scenario, where these newcomers lack historical data, and takes special measures to make accurate predictions despite the limited information available. This specialized attention to cold-start situations ensures that ARRQP can provide meaningful QoS predictions even for users or services with minimal or no prior usage history.

We introduced a scalable, real-time static QoS prediction framework (ARRQP) that effectively addresses multiple anomalies, including outliers, data sparsity, grey-sheep instances, and cold-start, ultimately enhancing prediction accuracy.

Our proposed framework comprises five key components: two anomaly detection blocks tasked with identifying grey-sheep users and services (GSU and GSS), and outliers, along with three closely integrated prediction blocks. The first prediction block is engineered to withstand outliers and data sparsity for regular users and services. Above this initial block, a dedicated prediction block is designed when either the target user, the service, or both lack sufficient data due to being new to the system. Moreover, if either the user or the service, or both, are identified as GSU and GSS, additional processing is applied to enhance the accuracy of predictions for this unique category. This layered approach ensures robustness against anomalies and adapts to diverse scenarios for effective QoS prediction. The primary contributions of this chapter are outlined as follows:

(i) **Novel core architecture**: We introduce a multi-layer multi-head graph convolution matrix factorization (MhGCMF) model as its core architecture, leveraging a graph-based representation to capture user-service interactions. Unlike traditional

matrix factorization, MhGCMF exploits the inherent graph structure of QoS data, utilizing a multi-layer design to capture hierarchical relationships. While existing graph-based learning frameworks [144] often suffer from over-smoothing and over-squeezing [73], our model mitigates these issues through residual connections, preserving feature expressiveness across layers. Additionally, MhGCMF employs multi-head graph convolution to extract diverse features, enhancing its ability to model complex user-service correlations. Furthermore, we introduce an outlier-resilient loss function, strengthening robustness against anomalous QoS data and ensuring reliability. Additionally, the synergy between contextual and QoS features, in addition to the spatial features automatically extracted by MhGCMF, significantly enhances the model's expressiveness in capturing the complex, higher-order association between user and services. This enhancement eventually leads to improved prediction performance.

(ii) **Sparsity-resilient grey-sheep detection and tailored-modeling for unique instances**: We present a sparsity-resilient method for detecting grey-sheep users or services characterized by unique attributes that challenge accurate QoS prediction through collaborative filtering alone. To address this, we introduce a dedicated processing module, informed by a quantitative distinction measure from the grey-sheep detection block. This tailored approach ensures more precise predictions for this distinct category of users or services.

(iii) **Addressing cold-start problem**: We address the cold-start problem by designing a separate model that prioritizes contextual features over similarity features.

(iv) **Comprehensive experiments and evaluation**: We conducted extensive experiments on the WS-DREAM 1 RT and TP datasets [1], along with a new smart transportation dataset featuring four QoS metrics, to evaluate the effectiveness of each ARRQP module and the overall framework performance.

The rest of this chapter is organized as follows. Section 4.1 then discusses the proposed solution framework in detail. The experimental results are analyzed in Section 4.2. Finally, Section 4.3 concludes this chapter.

4.1 Methodology

The standard static QoS prediction method employs Definition 4 for QoS modeling. However, ARRQP additionally focuses on enhancing the robustness and reliability of QoS prediction in realistic service environments. Specifically, ARRQP addresses the following research questions: (i) how to identify and mitigate the influence of anomalous entities, such as grey-sheep users and services, and abnormal QoS observations, (ii) how to perform accurate QoS prediction under sparse and unreliable interaction data, and (iii) how to enable reliable QoS prediction for cold-start users and services with limited historical information. Addressing these challenges enables ARRQP to provide resilient and dependable QoS prediction in the presence of sparsity, anomalies, and cold-start conditions.

In this section, we discuss our framework, ARRQP, for QoS prediction. ARRQP comprises two major anomaly detection blocks: (a) **G**rey-sheep user and service **D**etection block (GD), (b) **O**utlier **D**etection block (OD); and three major prediction blocks: (a) **S**parsity and **O**utlier **R**esilient **R**eal-time **Q**oS **P**rediction block (SORRQP), (b) **G**rey-sheep users and services **R**esilient **R**eal-time **Q**oS **P**rediction block (GRRQP), and (c) **C**old-start **R**esilient **R**eal-time **Q**oS **P**rediction block (CRRQP). The architectural overview of ARRQP is illustrated in Fig. 4.1 (c). In the following subsections, we discuss each of these blocks in detail.

4.1.1 SORRQP Block

SORRQP leverages graph convolution [67] to deal with the data sparsity. The graph convolutional network (GCN) takes privilege in graph architecture and effectively aggregates the node features through message passing between neighboring nodes. Graph architecture has more expressive power [67, 145] than most known representations of users and services. Therefore, we adopt graph convolution as the fundamental operation in the SORRQP architecture. Here, we propose a multi-layer multi-head graph convolution matrix factorization (MhGCMF) model for QoS prediction, shown in Fig.s 4.1 (a) and (b). Before discussing the further details of MhGCMF, we first define the

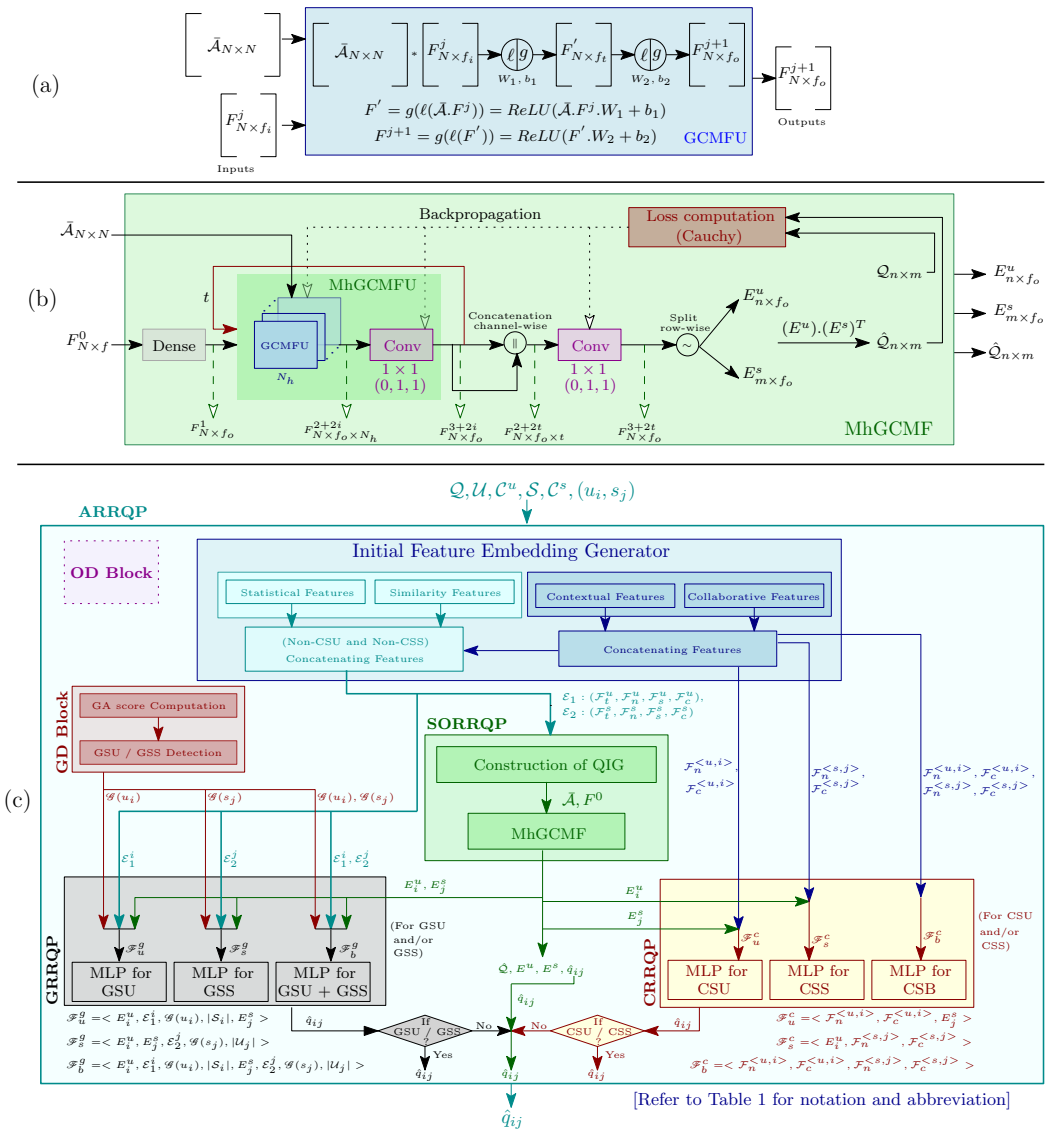


Figure 4.1: (a) Details of GCMFU, (b) Architecture for MhGCMF, and (c) Architecture for ARRQP

QoS Invocation Graph (QIG), which is a basic building block of graph convolution operation.

Definition 17 (QoS Invocation Graph (QIG)). *A QIG, $\mathcal{G} = (V_1 \cup V_2, E, \mathcal{E}_1 \cup \mathcal{E}_2)$ is a bipartite graph, where V_1 and V_2 are the set of vertices representing the set of users and services, respectively. An edge $e_{ij} = (v_i^1 \in V_1, v_j^2 \in V_2) \in E$ exists in $\mathcal{G}$ if $q_{ij} \neq 0$ in the QoS log matrix $\mathcal{Q}$. $\mathcal{E}_1$ and $\mathcal{E}_2$ represent the set of feature embedding for each node in V_1 and V_2, respectively.* ■

Table 4.1: Feature embedding for QIG

$min_i^u : min(\mathcal{Q}(i))$	$min_j^s : min(\mathcal{Q}^T(j))$
$max_i^u : max(\mathcal{Q}(i))$	$max_j^s : max(\mathcal{Q}^T(j))$
$\mu_i^u : mean(\mathcal{Q}(i))$	$\mu_j^s : mean(\mathcal{Q}^T(j))$
$med_i^u : median(\mathcal{Q}(i))$	$med_j^s : median(\mathcal{Q}^T(j))$
$\sigma_i^u : std\_dev(\mathcal{Q}(i))$	$\sigma_j^s : std\_dev(\mathcal{Q}^T(j))$

std_dev : standard deviation;$\mathcal{Q}(i)$: i^{th} row of $\mathcal{Q}$; $\mathcal{Q}^T$: Transpose of $\mathcal{Q}$

4.1.1.1 Construction of the Feature Embedding

Our initial feature embedding comprises a set of QoS features along with a set of contextual features. We now briefly discuss the details of this embedding.

4.1.1.1.1 QoS features. The QoS features include three different types of features, which are discussed below.

(i) Statistical features ($\mathcal{F}_{St}^u, \mathcal{F}_{St}^s$): To capture the self characteristics of each user u_i and service s_j, we compute 5 statistical features, as shown in Table 4.1. It may be noted that since $\mathcal{Q}(i)$ is a partially filled QoS invocation vector (QIV), the data sparsity affects the statistical features.

(ii) Collaborative features ($\mathcal{F}_{Q}^u, \mathcal{F}_{Q}^s$): They are derived from the QoS log matrix. We perform non-negative matrix decomposition [32] of $\mathcal{Q}$ to obtain the collaborative QoS features of u_i and s_j, each with dimension d_n.

(iii) Similarity features ($\mathcal{F}_{Si}^u, \mathcal{F}_{Si}^s$): They are also extracted from the QoS log matrix. We employ the cosine similarity metric (CSM) (refer to Definition 2.4) to obtain the similarity between pairwise users (u_i, u_k) and services (s_j, s_k), computed as:

$$\begin{aligned} CSM(u_i, u_k) &= \left(\mathcal{Q}(i) \cdot \mathcal{Q}(k)\right) / \left(\|\mathcal{Q}(i)\|_2 \, \|\mathcal{Q}(k)\|_2\right), \\ CSM(s_j, s_k) &= \left(\mathcal{Q}^T(j) \cdot \mathcal{Q}^T(k)\right) / \left(\|\mathcal{Q}^T(j)\|_2 \, \|\mathcal{Q}^T(k)\|_2\right) \end{aligned} \tag{4.1}$$

We use cosine similarity for similarity computation because it efficiently measures the direction of vectors, highlighting shared experiences regardless of engagement levels, and performs well with sparse, high-dimensional data. It is worth noting that the

similarity between two users and services is 0 if they do not have any common invocations. Therefore, in general, the similarity features remain sparse due to the sparsity of $\mathcal{Q}$. Integration of similarity features, however, significantly boosts the resilience, precision, and versatility of QoS prediction models. These features empower the model to quantify the connections between users and services through their past interactions, enabling the identification of behavioral patterns and service performance trends. Furthermore, depending on the number of users and services, the similarity feature vector may be high-dimensional. It may be noted that the length of the similarity feature vector for each user u_i and service s_j are n and m, respectively. Therefore, we employ two different auto-encoders [133] for the users and services separately to obtain the low-dimensional similarity feature vector of length d_s.

4.1.1.1.2 Contextual features ($\mathcal{F}_C^u, \mathcal{F}_C^s$). We utilize the contextual information associated with each u_i and s_j to prepare the contextual features. The user contextual features include the user ID, region, and autonomous system (AS) information, whereas the service contextual features comprise the service ID, service region, and service provider information. We use one-hot encoding for each contextual information. Here again, the dimension of the feature vector becomes large due to the high number of users and services. Therefore, we employ auto-encoders to obtain the contextual feature vector of length d_c.

Finally, all the above feature vectors are concatenated to construct the feature embedding

$$\mathcal{E}_1^i = (\mathcal{F}_{St}^{<u,i>} || \mathcal{F}_Q^{<u,i>} || \mathcal{F}_{Si}^{<u,i>} || \mathcal{F}_C^{<u,i>}) \in \mathcal{E}_1,$$

$$\mathcal{E}_2^j = (\mathcal{F}_{St}^{<s,j>} || \mathcal{F}_Q^{<s,j>} || \mathcal{F}_{Si}^{<s,j>} || \mathcal{F}_C^{<s,j>}) \in \mathcal{E}_2,$$

for each user u_i and service s_j, respectively. Here, $\mathcal{F}_t^{<u,i>}$ refers to the statistical features for u_i. The rest of them are denoted similarly. The length of the initial feature embedding (say, f) for each user and service is, therefore, $(5 + d_n + d_s + d_c)$.

The initial feature embedding matrix F^0 is defined in Eq. 4.2.

$$F^0 = (\rho_{ij}^0) \in \mathbb{R}^{(n+m)\times f}; \rho_i^0 = \begin{cases} \mathcal{E}_1^i \in \mathbb{R}^f, \text{ if } i \leqslant n \\ \mathcal{E}_2^{i-n} \in \mathbb{R}^f, \text{ otherwise} \end{cases} \tag{4.2}$$

The contextual features are less proficient than QoS features in capturing the intricate relationship among users and services, particularly in the context of QoS prediction. However, relying solely on QoS features often results in lower accuracy due to the sparse nature of the QoS log matrix. Therefore, we combine both types of features to create the initial feature embedding. In addition to the initial feature embedding, we incorporate spatial collaborative features obtained from the MhGCMF, which forms the core component of SORRQP. The graph architecture in the MhGCMF enables each user and service to integrate the neighbor information into its embedding, resulting in richer representations that facilitate effective QoS prediction. The primary goal of the MhGCMF is to learn the spatial features from the initial feature embedding of the users and services, to enhance the efficiency of QoS prediction. The GCMF Unit (GCMFU) is the central component for MhGCMF.

4.1.1.2 Architecture of GCMFU

GCMFU aggregates the neighborhood features of a node $v_i^k \in (V_1 \cup V_2),\ k \in \{1, 2\}$ from its neighbors, as modeled by the two equations shown in Fig. 4.1(a). Here, we leverage the adjacency matrix representation of $\mathcal{G}$ to aggregate the neighborhood features. To preserve the self-features of each node in the aggregation, we additionally set the diagonal entries of the adjacency matrix to 1. The modified adjacency matrix $\mathcal{A} \in \mathbb{R}^{N\times N}$ is defined in Eq. 4.3.

$$\mathcal{A} = (a_{ij}) \in \{0, 1\}^{N\times N}; a_{ij} = \begin{cases} 1, \text{ if } i = j \text{ or } e_{ij} \in E \\ 0, \text{ otherwise} \end{cases} \tag{4.3}$$

where, $N = n + m$. We then normalize $\mathcal{A}$, shown in Eq. 4.4, to avoid the scaling discrepancy caused by the varying degrees of the nodes in $\mathcal{G}$ because of the sparsity of $\mathcal{Q}$ and generate $\bar{\mathcal{A}}$. The diagonal degree matrix ($\mathcal{D}$), as defined in Eq. 4.5, is used for

the normalization. The normalization reduces the impact of the higher degree nodes in QoS prediction.

$$\bar{\mathcal{A}} = \mathcal{D}^{-1/2}\ \mathcal{A}\ \mathcal{D}^{-1/2} \tag{4.4}$$

$$\mathcal{D} = (d_{ij}) \in \mathbb{Z}^{N\times N};\quad d_{ij} = \begin{cases} \sum_{k=1}^{N} \mathcal{A}(i,k),\ \text{if } i = j \\ 0,\ \text{otherwise} \end{cases} \tag{4.5}$$

The GCMFU receives $\bar{\mathcal{A}}$ and a feature embedding matrix (say, F^j) as its input. It then performs a non-linear transformation on F^j with the help of a set of learnable parameters (refer to Fig. 4.1(a)) and produces the updated feature embedding matrix (say, F^{j+1}) by incorporating the spatial information of the neighborhood nodes.

4.1.1.3 Architecture of MhGCMF

Fig. 4.1(b) summarizes the architecture of the MhGCMF. Given the normalized adjacency matrix $\bar{\mathcal{A}}$ and the initial feature embedding matrix F^0, the objective of the MhGCMF is to learn the spatial collaborative features for each user and service for efficient QoS prediction.

In MhGCMF, F^0 first undergoes an initial transformation by passing through a dense layer, resulting in the transformed feature embedding F^1. Subsequently, F^1 and $\bar{\mathcal{A}}$ are directed to the multi-head GCMFU (i.e., MhGCMFU) block. MhGCMFU comprises N_h number of GCMFUs followed by a 1×1 convolution layer. The "multi-head" in MhGCMFU refers to the presence of multiple GCMFUs. The outputs of these multiple GCMFUs are then concatenated channel-wise, and the resulting tensor passes through a 1×1 convolutional layer. This entire process is repeated t times, where the output of MhGCMFU is fed back into itself. The output generated by each MhGCMFU block is once again concatenated channel-wise and then forwarded to another 1×1 convolutional layer. For each 1×1 convolutional layer, there is a corresponding tuple indicating (padding, stride, and number of filters), as illustrated in Fig. 4.1 (b). The output of the final 1×1 convolutional layer is split row-wise to produce the user and service embedding matrices. These matrices are subsequently multiplied to generate the predicted QoS log matrix. We train the MhGCMF for each

user-service pair $(u_i, s_j) \in \mathcal{U} \times \mathcal{S}$, such that $q_{ij} \neq 0$ using Cauchy Loss Function [136], as shown in Eq. 4.6.

$$\mathcal{C}_{\mathcal{L}} = \sum_{q_{ij} \neq 0} \ln\left(1 + \frac{||q_{ij} - \hat{q}_{ij}||^2}{\gamma^2}\right) \tag{4.6}$$

where, γ is a hyper-parameter that can be tuned externally, and $\hat{q}_{ij}$ is the predicted QoS value. The Cauchy loss function endows SORRQP with outlier resilience characteristics. However, SORRQP alone cannot address the other issues, such as grey-sheep and cold-start. In the next section, we address the issue of grey-sheep.

4.1.2 Grey-sheep Detection Block (GD)

Collaborative Filtering (CF) highly relies on the collaborative relationship between users and services and operates on the premise that similar users and services exist within the system. However, it is worth noting that there can be a small group of users or services with QoS patterns that deviate from the norm, making them distinct from the majority. These users and services are often referred to as *grey-sheep*, representing an anomalous subset within the user or service community [86]. The existence of grey-sheep users or services may occasionally lead to substantial inaccuracies in QoS predictions.

In this context, our approach begins with addressing the task of identifying grey-sheep Users and Services (GSU and GSS). Subsequently, we tackle the challenge of QoS prediction for these identified GSUs and/or GSSs. The GD block within ARRQP primarily focuses on identifying the instances of grey-sheep within the QoS data.

To capture the distinct QoS pattern exhibited by a user and service, here, we introduce two concepts: *reliability score* ($Rel(\cdot)$) and *Grey-sheep Anomaly (GA) score* [86] for each user and service. These two scores together aid in determining whether a particular user and service qualify as a GSU and GSS.

Definition 18 (Reliability Score)**.** *The reliability score of a user $u_i \in \mathcal{U}$ and service $s_j \in \mathcal{S}$, denoted by $Rel(u_i)$ and $Rel(s_j)$, respectively, are defined as:*

$$Rel(u_i) = 1 - \left(\frac{\sigma_i^u - \min\limits_{u_k \in \mathcal{U}}(\sigma_k^u)}{\max\limits_{u_k \in \mathcal{U}}(\sigma_k^u) - \min\limits_{u_k \in \mathcal{U}}(\sigma_k^u)}\right) \tag{4.7}$$

$$Rel(s_j) = 1 - \left(\frac{\sigma_j^s - \min\limits_{s_k \in \mathcal{S}}(\sigma_k^s)}{\max\limits_{s_k \in \mathcal{S}}(\sigma_k^s) - \min\limits_{s_k \in \mathcal{S}}(\sigma_k^s)} \right) \tag{4.8}$$

σ_i^u and σ_j^s represent the standard deviation of the QIVs of u_i and s_j, respectively (refer to Table 4.1). ■

It may be noted from Eq. 4.7 that a user or service is said to be more reliable than another if the standard deviation of the QIV of the former is lower than that of the latter. We now define the GA score in Definition 19.

Definition 19 (Grey-sheep Anomaly (GA) Score)**.** *The GA score of a user $u_i \in \mathcal{U}$ and service $s_j \in \mathcal{S}$, denoted by $\mathscr{G}_A(u_i)$ and $\mathscr{G}_A(s_j)$, respectively, are defined as:*

$$\mathscr{G}_A(u_i) = \left(\sum_{s_j \in \mathcal{S}_i} \left(|q_{ij} - \mu_i^u - \bar{\mu}_j^s| \times Rel(s_j) \right) \right) / |\mathcal{S}_i| \tag{4.9}$$

$$\mathscr{G}_A(s_j) = \left(\sum_{u_i \in \mathcal{U}_j} \left(|q_{ij} - \mu_j^s - \bar{\mu}_i^u| \times Rel(u_i) \right) \right) / |\mathcal{U}_j| \tag{4.10}$$

$$\bar{\mu}_j^s = \left(\left(\sum_{u_i \in \mathcal{U}_j} q_{ij} \right) - \max_{u_i \in \mathcal{U}_j}(q_{ij}) - \min_{u_i \in \mathcal{U}_j}(q_{ij}) \right) / \left(|\mathcal{U}_j| - 2 \right) \tag{4.11}$$

$$\bar{\mu}_i^u = \left(\left(\sum_{s_j \in \mathcal{S}_i} q_{ij} \right) - \max_{s_j \in \mathcal{S}_i}(q_{ij}) - \min_{s_j \in \mathcal{S}_i}(q_{ij}) \right) / \left(|\mathcal{S}_i| - 2 \right) \tag{4.12}$$

$\mathcal{S}_i$ be the set of services invoked by u_i. $\mathcal{U}_j$ be the set of users invoked s_j. μ_i^u and μ_j^s represent the mean of the QIVs of u_i and s_j, respectively. ■

It may be noted from the above equations that the GA score of each user u_i (service s_j) is computed over its respective QIV, denoted by $\mathcal{Q}(i)$ ($\mathcal{Q}^T(j)$). For each invocation q_{ij} ($\neq 0$) made by u_i (for s_j), we compute the deviation with respect to the mean of QIV of u_i (s_j) and the centralized mean of s_j (u_i). The average of these weighted deviations, computed over the QIV of u_i (s_j), is referred to as the GA score of u_i (s_j). The reliability score of the corresponding service is used as the weight in this computation. Notably, if a service has a high reliability score, the corresponding deviation carries more weight than one associated with a service with a lower reliability

score. It may be noted that the GA score of a user only depends on the set of services it invokes. Therefore, the GA score is not affected by the data sparsity. We now define the Grey-sheep user (GSU) and service (GSS).

Definition 20 (Grey-sheep user (GSU))**.** *A user $u_i \in \mathcal{U}$ is called GSU, if $\mathscr{G}(u_i)$ is more than a given threshold $\tau^u_{\mathscr{G}}$.* ■

Definition 21 (Grey-sheep service (GSS))**.** *A service $s_j \in \mathcal{S}$ is called GSS, if $\mathscr{G}(s_j)$ is more than a given threshold $\tau^s_{\mathscr{G}}$.* ■

$\tau^u_{\mathscr{G}}$ and $\tau^s_{\mathscr{G}}$ are hyper-parameters. We consider $\tau^u_{\mathscr{G}} = \mu^u_{\mathscr{G}} + c * \sigma^u_{\mathscr{G}}$ ($\tau^s_{\mathscr{G}} = \mu^s_{\mathscr{G}} + c * \sigma^s_{\mathscr{G}}$), where $\mu^u_{\mathscr{G}}$ and $\sigma^u_{\mathscr{G}}$ ($\mu^s_{\mathscr{G}}$ and $\sigma^s_{\mathscr{G}}$) are the mean and standard deviation of the GA scores of users (services), respectively. $c \in \mathbb{R}_{>0}$ is a hyper-parameter, which can be tuned externally.

Once the GD block identifies the GSU and GSS, the next step is to predict the QoS values for these identified GSUs and GSSs using our next prediction block, GRRQP.

4.1.3 GRRQP Block

The GRRQP is primarily designed to provide QoS predictions for GSU and GSS (refers to GRRQP block in Fig. 4.1(c)). The rationale behind introducing the GRRQP block stems from the recognition that grey-sheep users and services exhibit markedly distinct QoS patterns compared to the majority of users and services. Consequently, relying extensively on collaborative features, as employed in SORRQP, may not be adequate for achieving higher QoS predictions for GSU and GSS. Furthermore, the prediction of QoS values for GSU and GSS in conjunction with other users and services may not be advisable, given that GSU and GSS possess distinct characteristics that set them apart from the rest. Therefore, here, we design an additional module for the QoS prediction tailored specifically for GSU and GSS.

The GRRQP is an improvisation of SORRQP, featuring supplementary MLPs (Multi-Layer Perceptrons) in addition to the MhGCMF. Here, we have three MLPs designed specifically for (a) Non-greysheep users and GSSs, (b) GSUs and non-greysheep services, and (c) GSUs and GSSs. The feature vector for each MLP is constructed by

concatenating the user and service features. If the user u_i is a non-greysheep user, the embedding vector E_i^u (i.e., i^{th} row of E^u) obtained from MhGCMF is used as its features. However, if u_i is a grey-sheep user, its feature is generated by concatenating the following components: (a) The embedding vector E_i^u obtained from MhGCMF, (b) The initial feature embedding vector $\mathcal{E}_1^i$ as it captures the user's own unique characteristics, (c) The GA score $\mathscr{G}_A(u_i)$ as it indicates the abnormality of the user, and in turn, shows the uniqueness of the user, and (d) The number of invocations of u_i. The service features are also generated similarly. The MLPs are trained for the user-service pair in the respective categories for which $q_{ij} \neq 0$.

4.1.4 CRRQP Block

The term *cold-start* pertains to a scenario in which a new user and service is introduced into a system. This situation poses a significant challenge for QoS prediction due to the lack of available data. The SORRQP system is ill-suited to handle the cold-start scenario for the following reasons: (a) The similarity and statistical features in the initial feature embedding cannot provide meaningful information for the newly added user and service due to the absence of historical data. (b) The newly introduced user and service forms an isolated node within the QIG, indicating no interactions among other users and services. These isolated nodes lack the ability to capture spatial features effectively. As a result, this limitation poses a challenge for SORRQP in making accurate predictions. Therefore, we propose another prediction block, CRRQP, to address the above issue. CRRQP is an enhancement over SORRQP.

In addition to MhGCMF, CRRQP is also equipped with three MLPs to predict the QoS of the following target user-service pairs (refer to CRRQP block in Fig. 4.1 (c)). (a) Cold-start User (CSU): Here, the target user is new. However, the service has sufficient past data. (b) Cold-start Service (CSS): In this case, the target service is new, while the user has historical data. (c) Cold-start user and service Both (CSB): This scenario involves both the user and service being newly introduced entities without any historical data available for either of them.

The three MLPs are trained separately with three different sets of input features.

The input features are constructed by concatenating the user and the service features. If the user and service u_i and s_j is not a newly added, the embedding vector E_i^u and E_j^s obtained from MhGCMF is used as its features.

However, if a new user and service is added, its feature is generated by concatenating its contextual features with the collaborative features obtained using non-negative matrix decomposition. While the collaborative features provide a global overview of the QoS data and assist in the prediction process, it is crucial to acknowledge that the newly added user and service may lack statistical data or similarity features due to the absence of past interactions. Additionally, spatial features are unavailable for these entities as they represent isolated nodes in the QIG without any prior interactions.

4.1.5 Outlier Detection Block

We employ an unsupervised learning algorithm based on the isolation forest (*i*Forest) algorithm [109] to detect the outliers from scratch. *i*Forest algorithm explicitly isolates the outliers rather than profiling the inliers, in contrast to the density-based approaches for outlier detection. The strength of the *i*Forest approach is its effectiveness over different benchmarks and high scalability. It uses an ensemble of isolation trees (*i*Tree), whose branch grows iteratively; this way, it isolates every single data point. Finally, in each *i*Tree, the outliers are instances that are isolated near the root of the tree and have comparatively short average path lengths, consecutively, inliers are isolated to the far end of the *i*Tree.

After identifying the outliers, we proceed to eliminate them from the dataset, with the ratio of removed outliers denoted by λ (e.g., $\lambda = 0.1$ refers to 10% of the total dataset considered as the outlier). This step is essential to evaluate the performance of our framework without outliers.

4.1.6 Overall Architecture and Computational Complexity

Fig. 4.2 outlines overall ARRQP framework. Upon receiving a user-service pair, the it first extracts initial features. For regular pairs (neither grey-sheep nor new), it com-

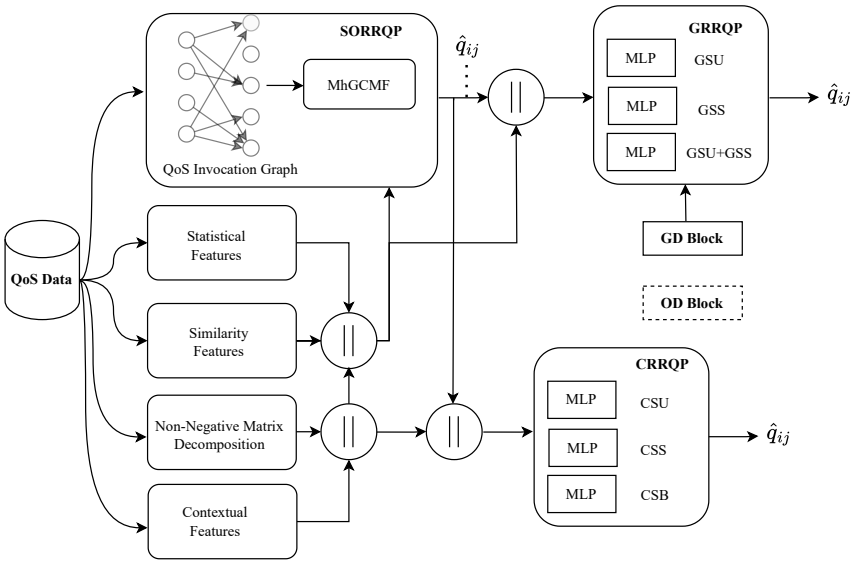


Figure 4.2: Overall architecture ARRQP

bines statistical, similarity, contextual, and collaborative features, which are processed by the SORRQP block for QoS prediction. For new users or services, the CRRQP block predicts QoS using contextual and collaborative features for the new entity, while embeddings from the SORRQP block are used for regular entities. If either the user or service is identified as a grey-sheep by the GD block, the GRRQP block integrates initial features, SORRQP embeddings, and the GA score to predict QoS.

The computational complexity of ARRQP arises primarily from two components: the initial feature extraction and the QoS prediction modules. Feature extraction involves contextual, statistical, similarity-based, and NMD features, where similarity features ($O(n^2m + m^2n)$) and NMD ($O(nmd)$) contribute most significantly, here, n and m denote the number of users and services respectively, and d is the number of latent features. Among the prediction modules, the SORRQP module incurs the highest complexity, especially from its multi-head graph convolutional framework (MhGCMF), dominated by $O(N^2 f_0 N_h t)$, where $N = n + m$ is the total number of nodes in the QoS Invocation Graph, f_0 is the embedding size, N_h is the number of attention heads, and t is the number of convolutional layers. GRRQP and CRRQP exhibit moderate complexity $O(L_1 fd)$, where L_1 is the number of layers in each MLP,

f is the feature dimension, and d is the output size per layer. The anomaly detection module (GD) maintains low linear complexity $O(e)$, with e being the number of QoS invocations (edges in the bipartite graph). Overall, the major factors contributing to increased complexity are the high dimensionality of input features, pairwise similarity computations, and graph-based multi-head convolution operations.

4.2 Experimental Results

We implemented our framework using TensorFlow 2.6.2 with Python 3.6.9. All experiments were executed on the Linux-5.4.0-133-generic-x86_64-with-Ubuntu-18.04-bionic with Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz x86_64 architecture, 16 Core(s), 128 GB RAM, with the following cache L1d: 32K, L1i:32K, L2:256K, L3:16384K.

4.2.1 Experimental Setup

4.2.1.1 Datasets

We primarily used the publicly available WSDREAM-1 [1] datasets demonstrated in Table 2.3 in Sec. 2.3. In addition, we evaluate our framework on a new gRPC dataset as described below.

4.2.1.2 Proposed gRPC Dataset

We constructed a small dataset in the smart transportation domain, focusing on autonomous vehicles (AVs), using the gRPC framework[1], a high-performance, open-source RPC system that enables load balancing, authentication, and seamless communication across devices, applications, and backend services. Inspired by Pylot [146], which introduces a modular AV pipeline to examine the effects of QoS parameters like latency and reliability, we identified five essential tasks for AV operation: Object Detection, Object Tracking, Prediction, Planning, and Control.

[1]https://grpc.io/

Table 4.2: Details of a new gRPC service dataset

Contextual Info	User		Service	
Quantity	57		150	
AS	16		1	
City	27		1	
Country	3		1	
Region	13		1	
Statistics	**Reliability (RE)**	**Cost (CT)**	**Latency (LT)**	**Power (PW)**
Range (min-max)	44.61%-99.99%	4.6000-4.6000	0.0484-151.8636	17.2100-92.7500
Mean, Median	68.38%, 63.28%	11.2214, 11.2000	3.7787, 1.1787	56.5041, 66.2100
Standard Deviation	11.26	1.9538	9.5216	20.0000

Table 4.3: Details of various hyperparameter used in experiments

Parameters		RT-5	RT-10	RT-15	RT-20	TP-5	TP-10	TP-15	TP-20
c, λ, t		2, 0.1, 2							
d_n, d_s, d_c		50, 50, 50							
N_h	MAE	5	1	4	3	5	4	5	5
	RMSE	5	1	2	2	5	7	8	6
γ	MAE	0.05	0.25	0.25	0.25	10	10	50	10
	RMSE	250	10	100	10	10	50	50	100

To study QoS provisions for these tasks, we implemented 150 machine learning algorithms and deployed them as gRPC-based services. These services were hosted on five servers at our institute, and data was collected from 57 distributed users across 27 cities in India, Taiwan, and the US. We recorded data for four QoS parameters: *(i) Reliability (RE)*, defined as the task accuracy achieved by services and measured in percentage, *(ii) Latency (LT)*, measured in seconds, indicating end-to-end response time, *(*iii) Cost (CT), calculated as the ratio of memory consumed during service execution to the total available memory, and *(*iv) Power (PW) represents the energy consumption measured in watts.

Table 4.2 summarizes gRPC dataset details across these parameters.

4.2.1.3 Parameter Configurations

To validate the performance of ARRQP, we used four different (training, testing) split-ups: $(5\%, 95\%)$, $(10\%, 90\%)$, $(15\%, 85\%)$, and $(20\%, 80\%)$. In this chapter, we use the notation RT-x (TP-x) to denote the (training, testing) split up for $(x\%, (100 - x)\%)$.

In other words, for RT-x, we used x% data of the given response time values for training, while the rest was used for testing. Moreover, we used 20% of the training data as the validation dataset for our experiment. We repeated each experiment 10 times and reported the average value. The dataset is randomly split into training, testing, and validation sets in each experiment.

In all our experiments, unless specifically stated otherwise, we maintained consistent usage of the parameters with values as listed in Table 4.3.

Table 4.4: Comparative study of ARRQP on WSDREAM-1 RT dataset

Anomaly Addressed	Methods	Loss Function	MAE				RMSE			
			5	10	15	20	5	10	15	20
None	UPCC [6]	-	0.7695	0.7201	0.6521	0.5921	1.6940	1.6101	1.4959	1.3993
	IPCC [7]	-	0.7326	0.7125	0.6783	0.6503	1.6346	1.5439	1.4104	1.3109
	WSRec [8]	-	0.6794	0.6211	0.6037	0.6020	1.4884	1.4265	1.3684	1.3515
	PG (%)		**46.46**	**49.49**	**54.16**	**56.32**	**16.83**	**19.69**	**20.51**	**23.41**
$\mathbb{S}+\mathbb{C}$	NMF [32]	MSE	0.6182	0.6040	0.5990	0.5982	1.5746	1.5494	1.5345	1.5531
	PMF [35]	MSE	0.5678	0.4996	0.4720	0.4492	1.4735	1.2866	1.2163	1.1828
	GeoMF [12]	MSE	0.5305	0.4827	0.4495	0.4366	1.3152	1.2190	1.1742	1.1528
	CNMF [147]	MSE	0.5289	0.4713	0.4316	0.4136	1.3053	1.2373	1.1360	1.1161
	CSMF [39]	MSE	0.5286	0.4557	0.4225	0.4180	1.3473	1.2315	1.1780	1.1210
	LMF-PP [66]	MSE	0.5285	0.4725	0.4472	0.4260	1.3410	1.2419	1.2102	1.1625
	JCF [148]	MSE	0.5132	0.4665	0.4504	0.4365	1.3328	1.2505	1.1854	1.1802
	NDMF [47]	MSE	0.4880	0.4304	0.3845	0.3665	1.3495	1.2349	1.1569	1.1294
	EFMPred [149]	MSE	0.4460	0.3750	0.3630	0.3480	1.3530	1.2810	1.2530	1.2360
	MSDAE [83]	MSE	0.4267	0.3640	0.3434	0.3275	1.2853	1.2436	1.1695	1.1283
	LAFIL [81]	MSE	0.3880	0.3664	0.3460	0.3268	1.2674	1.1926	1.1106	1.0923
	PG (%)		**6.26**	**13.82**	**19.42**	**22.58**	**2.32**	**3.94**	**2.06**	**8.95**
$\mathbb{S}+\mathbb{C}+\mathbb{O}$	DCALF [150]	MSE	0.5127	0.4544	0.4346	0.4246	1.3731	1.2450	1.2001	1.1759
	FHC-DQP [90]	MAE	0.5100	0.4340	0.3950	0.3380	1.4000	1.3160	1.2360	1.2050
	NCF [151]	Huber	0.4400	0.3850	0.3720	0.3620	1.3250	1.2830	1.2530	1.2050
	SPP+LLMF [64]	MAE	0.4350	0.3700	0.3550	0.3360	1.3340	1.2500	1.1920	1.1740
	DNM [92]	MAE	0.4125	0.3726	0.3678	0.3575	1.3463	1.2673	1.2490	1.2298
	MM-DNN [51]	MAE	0.4102	0.3392	0.3261	0.3117	1.2165	1.0835	1.0665	1.0321
	LDCF [13]	Huber	0.4020	0.3670	0.3450	0.3310	1.2770	1.2330	1.1690	1.1380
	DCLG [52]	Huber	0.3850	0.3460	0.3260	0.3140	1.2630	1.1850	1.1430	1.1200
	FRLN [91]	MAE	0.3790	0.3440	0.3220	0.3090	1.3060	1.2380	1.2010	1.1750
	QoSGNN [55]	MAE	0.3770	0.345	-	-	1.3350	1.2760	-	-
	HSA-Net [87]	MAE	0.3670	0.3270	0.3070	0.2950	1.2490	1.1660	1.1260	1.0910
	NCRL [59]	MAE	0.3589	0.3385	-	-	1.2694	1.2252	-	-
	PG (%)		**-1.34**	**4.06**	**9.86**	**14.24**	**0.89**	**1.75**	**3.40**	**8.85**
$\mathbb{S}+\mathbb{GS}+\mathbb{O}+\mathbb{C}$	**ARRQP**	**Cauchy**	**0.3637**	**0.3137**	**0.2767**	**0.2530**	**1.2379**	**1.1456**	**1.0877**	**0.9945**

4.2.2 Performance Analysis

We now present the performance analysis of our experimental results.

Table 4.5: Comparative study of ARRQP on WSDREAM-1 TP dataset

Anomaly Addressed	Methods	Loss Function	MAE				RMSE			
			5	10	15	20	5	10	15	20
None	UPCC [6]	-	27.5760	23.0130	20.6800	19.2940	63.7540	56.6210	51.9550	47.6510
	IPCC [7]	-	26.8640	26.0590	25.8790	24.2610	65.7900	61.1320	59.1500	54.6970
	WSRec [8]	-	26.7330	22.6940	20.5470	18.9640	63.4980	56.4590	51.7890	46.9510
PG (%)			**50.40**	**52.53**	**52.50**	**53.97**	**36.53**	**34.39**	**32.48**	**29.73**
$\mathbb{S}+\mathbb{C}$	CSMF [39]	MSE	28.3060	26.5850	25.6250	21.7420	72.7040	70.0210	68.6180	61.6790
	NMF [32]	MSE	25.7529	17.8411	15.8939	15.2516	65.8517	53.9896	51.7322	48.6330
	GeoMF [12]	MSE	24.7465	22.4728	17.7908	16.2852	57.7842	49.2456	45.3255	43.9545
	PMF [35]	MSE	19.9034	16.1755	15.0956	14.6694	54.0508	46.4439	43.7957	42.4855
	EFMPred [149]	MSE	19.8460	17.4610	17.0460	15.8390	73.8420	65.6800	67.0930	61.8030
	LRMF [77]	MSE	19.1090	15.9494	14.5974	13.9206	58.0719	48.2718	44.0682	41.7880
	LMF-PP [66]	MSE	18.3091	15.9125	14.7450	14.1033	51.7765	46.1418	42.9927	41.4084
	NAMF [152]	MSE	18.0836	15.9808	14.6661	13.9386	52.8658	44.0788	43.0206	40.7481
	NIMF [37]	MSE	17.9297	16.0542	14.4363	13.7099	51.6573	45.9409	43.1596	41.1689
	NDMF [47]	MSE	16.3818	13.9317	12.5043	11.7204	50.9612	43.9095	42.5319	39.9431
	LAFIL [81]	MSE	13.9875	12.2119	11.3761	10.9294	48.5321	42.5069	40.2270	38.6575
PG (%)			**14.77**	**11.79**	**14.20**	**20.13**	**16.96**	**12.85**	**13.89**	**14.66**
$\mathbb{S}+\mathbb{C}+\mathbb{O}$	DNM [92]	MAE	18.4903	16.2861	15.1406	14.7933	63.2993	55.0821	49.3940	48.4284
	DCALF [113]	MSE	17.6576	15.3595	14.3936	13.6697	51.4123	45.9013	42.6235	41.2194
	FHC-DQP [90]	MAE	17.2660	14.3470	13.4950	12.6380	51.5200	46.6050	44.9160	42.1970
	QoSGNN [55]	MAE	16.2550	13.9460	-	-	54.2060	47.9550	-	-
	MM-DNN [51]	MAE	16.1658	13.3631	12.4415	11.7019	50.0102	42.4129	39.1015	37.2234
	NCF [151]	Huber	15.4680	13.6160	12.2840	11.8330	49.7030	46.3040	42.3170	41.2630
	SPP+LLMF [64]	MAE	15.3820	13.6540	11.9040	11.0890	51.5660	45.6810	41.6310	39.5340
	FRLN [91]	MAE	14.9400	12.5700	11.3700	11.0600	51.6200	44.6700	40.8700	39.6000
	LDCF [13]	Huber	13.8440	12.3820	11.2700	10.8410	47.3590	43.4820	39.8130	38.9980
	NCRL [59]	MAE	13.4900	11.8400	-	-	46.2500	41.4500	-	-
	DCLG [52]	Huber	12.9280	11.3040	10.4060	9.9360	43.9690	39.9600	37.6370	36.1760
PG (%)			**-1.97**	**4.70**	**6.20**	**12.15**	**8.34**	**7.30**	**7.97**	**8.80**
$\mathbb{S}+\mathbb{GS}+\mathbb{O}+\mathbb{C}$	**ARRQP**	**Cauchy**	**13.1839**	**10.7723**	**9.7604**	**8.7285**	**40.3005**	**37.0415**	**34.9693**	**32.9910**

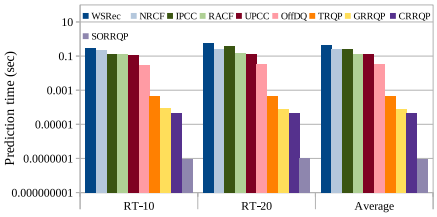


Figure 4.3: Comparative study of ARRQP on prediction time

Table 4.6: Prediction time for different blocks of ARRQP

Method	Avg. Train time (sec)	Prediction time (sec)
SORRQP	368	$5.008\ E^{-08} \pm 4.597\ E^{-08}$
GRRQP	540	$8.335\ E^{-06} \pm 1.639\ E^{-06}$
CRRQP	492	$4.540\ E^{-06} \pm 4.527\ E^{-07}$

4.2.2.1 Comparison with SOTA Methods

We compared ARRQP with 26 and 23 major state-of-the-art (SOTA) methods on RT and TP datasets, respectively. Table 4.5 shows the performance of ARRQP on RT and TP datasets for different training densities in terms of prediction accuracy. Here are our observations on the performance of ARRQP:

(i) Comparison of prediction accuracy with SOTA: As observed from Table 4.4 and Table 4.5, the prediction error started decreasing as the methods started handling more anomalies. It is worth noting that ARRQP outperformed all the SOTA methods reported in Table 4.4 and Table 4.5 since it addressed more variations of anomalies compared to contemporary methods. The improvements of ARRQP over the other methods for each case (the methods are divided based on the anomalies addressed by them) are shown in the tables in bold.

(ii) Comparison of prediction accuracy between ARRQP and DCLG/NCRL: It may be noted from Table 4.5, the performance of ARRQP degraded by -1.97% compared to DCLG [52] in terms of MAE on TP-5. However, the performance of ARRQP improved by 8.34% for the same in terms of RMSE. This shows that even though ARRQP increased the overall errors compared to DCLG, ARRQP was able to reduce the large prediction errors significantly. For the rest of the other cases, however, ARRQP outperformed DCLG in terms of MAE and RMSE. A similar observation can be made for NCRL on the RT dataset, as shown in Table 4.4.

(iii) Change of prediction accuracy with training density: As observed from Table 4.4 and Table 4.5, the performance of ARRQP improved with the increase in the training density. This is because, as the density increases, the number of neighbors of each node in the QoS prediction graph increases, which allows each user and service to have additional spatial collaborative information in the feature vector.

(iv) Training time of ARRQP: The training of ARRQP is performed in offline mode as discussed in Section 4.1. The training time for ARRQP varied from 1.8 to 21.75 minutes, depending on the number of heads used in GCMF, which varied from 1 to 8 in our experiments.

(v) Performance of ARRQP in terms of prediction time: Fig. 4.3 provides a comparative analysis of the prediction time between ARRQP and several SOTA methods (such as WSRec [8], NRCF [28], IPCC [7], RACF [29], UPCC [6], OffDQ [58], and TRQP [108]), focusing on the average prediction time for one target user-service

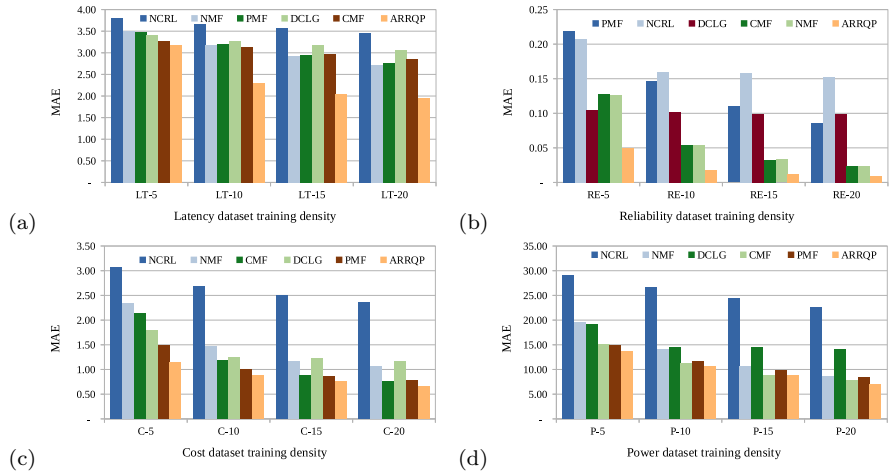


Figure 4.4: Comparison with SOTA methods on gRPC services dataset with four QoS parameters (a) Latency, (b) Reliability, (c) Cost, and (d) Power

pair. The results indicate that our framework exhibits reasonable responsiveness compared to other methods, as depicted in Fig. 4.3. Table 4.6 shows the training and prediction time of SORRQP, GRRQP, and CRRQP. While the training time of our framework is notably high, the prediction time remains quite reasonable, rendering ARRQP suitable for real-time systems since training is conducted offline. It is worth noting that the prediction time for CRRQP and GRRQP is higher than SORRQP due to the additional processing involved. However, only a tiny fraction of user-service pairs trigger this extra computation, with most relying on SORRQP. Consequently, the average prediction time of ARRQP remains comparable to that of SORRQP.

(vi) *Performance of ARRQP on the new dataset*: We evaluate the performance of ARRQP on the four QoS parameters in our dataset under varying sparsity levels ranging from 5% to 20% in 5% increments. For comparison, we selected two classical matrix factorization methods, NMF [32] and PMF [35], one outlier-resilient method, CMF [42], and two deep learning-based methods [52, 59], which demonstrated performance close to ARRQP on the WSDREAM datasets. As shown in Fig.s 4.4 (a)-(d), ARRQP consistently outperformed these methods.

Table 4.7: Performance of SORRQP after removal of outliers

Methods	RT-10		RT-20		TP-10		TP-20	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LLMF [64]	0.4041	0.6120	0.4037	0.6106	16.5509	33.8889	13.1105	27.9648
DALF [153]	0.3955	0.7466	0.3439	0.6779	13.1968	27.8531	11.9619	26.0299
CAP [16]	0.3603	0.6439	0.3521	0.6640	16.4269	32.9558	16.3125	32.9334
TAP [15]	0.3385	0.5512	0.2843	0.4985	22.1419	43.4987	19.8273	40.9533
TRQP [108]	0.2540	-	0.2520	-	10.5760	-	9.5660	-
OffDQ [58]	0.2000	-	0.1800	-	9.1600	-	8.6700	-
CMF [42]	0.1762	0.3705	0.1524	0.3599	8.4573	24.9137	7.2501	20.8927
SORRQP	**0.0930**	**0.3556**	**0.0794**	**0.2710**	**4.9652**	**18.3332**	**4.0876**	**16.6755**
PG (%)	**47.21**	**4.02**	**47.90**	**24.69**	**41.29**	**26.41**	**43.62**	**20.18**

* Improvement (*PG*) of SORRQP over the second best method highlighted in blue

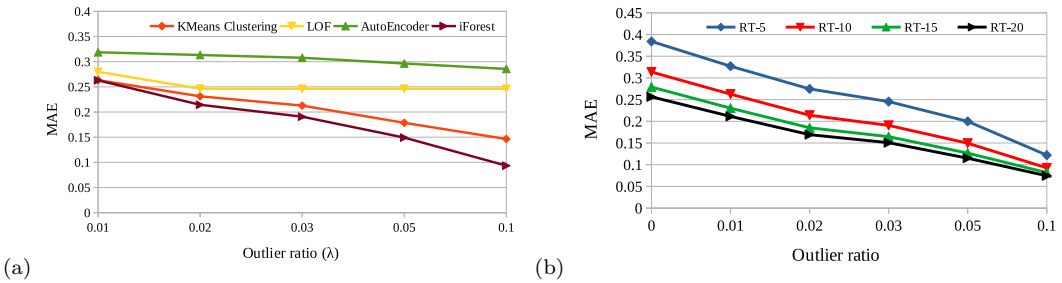
4.2.2.2 Performance of ARRQP on Addressing Anomalies

This subsection analyzes on various anomaly detection mechanisms and the performance of ARRQP in dealing with those anomalies.

4.2.2.2.1 Impact of Outlier Detection Method. Table 4.7 shows the performance of SORRQP after the removal of 10% outliers (i.e., $\lambda = 0.1$). Here, we consider the SOTA methods after removing 10% or more data with anomalies. As observed from the final row of Table 4.7, SORRQP outperformed all the SOTA methods in terms of prediction accuracy by a significant margin.

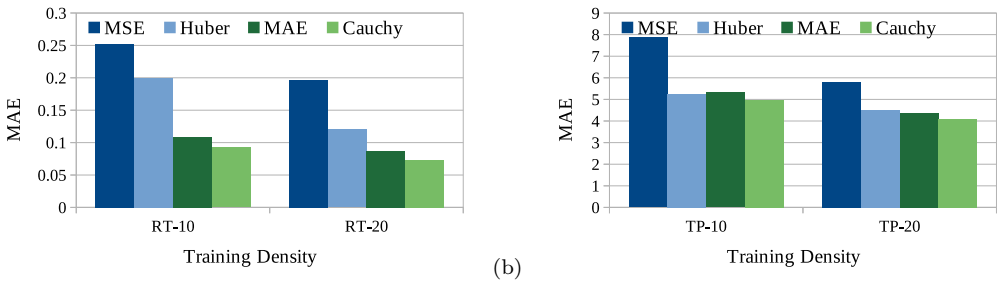
Fig. 4.5 (a) presents a comparative analysis of various outlier detection methods, revealing that i-Forest consistently outperformed K-Means [154], Local Outlier Factor (LOF) [155], and AutoEncoder (AE) [133] across all training densities and datasets. Its tree-based isolation mechanism makes it both scalable and robust, enabling superior outlier detection. In contrast, K-Means is moderately effective but sensitive to extreme values, while LOF struggles due to its reliance on local density, missing a significant portion of outliers. Although AE-based method effectively models complex patterns, its reliance on reconstruction error limits its outlier detection capability. Overall, i-Forest proved to be the most reliable method, justifying its use in uncovering the true performance of ARRQP.

Fig.4.5 (b) shows the impact of the outlier detection algorithm used on the RT dataset. Here, we show the performance of our framework with the removal of



(a) (b)

Figure 4.5: (a) Comparative analysis of various outlier detection methods on RT-10, (b) Analysis of outliers on RT dataset



(a) (b)

Figure 4.6: Analysis of loss functions (a) RT and (b) TP datasets

$1\%, 2\%, 3\%, 5\%$ and 10% outliers. As observed from Fig. 4.5 (b), the performance of ARRQP (in terms of MAE) improved as more outliers were detected and removed, which shows the efficiency of the outlier detection algorithm.

4.2.2.2.2 Impact of Loss Function to Handle Outliers. Fig.s 4.6 (a) and (b) show the impact of the loss functions on the performance of the ARRQP on RT and TP datasets. The MSE is an outlier-sensitive loss function; therefore, it performed worst. MAE, Huber loss and Cauchy loss can deal with the outliers. In our framework, we adopt the Cauchy loss to train our model since it outperformed all other loss functions, as evident from Fig.s 4.6 (a) and (b).

Table 4.8: Number of (GSU, GSS) for different c

$c = 1$		$c = 2$		$c = 3$	
RT	TP	RT	TP	RT	TP
(35, 254)	(24, 304)	(25, 150)	(7, 233)	(8, 101)	(4, 176)

Table 4.9: Performance (MAE) of GRRQP over SORRQP

Dataset	SORRQP		GRRQP		$\mathcal{I}$ (%)	
	Case-1	Case-2	Case-1	Case-2	Case-1	Case-2
RT-10	1.7507	0.9278	1.7375	0.8857	**0.75**	**4.54**
RT-20	1.4828	0.6441	1.4568	0.5962	**1.75**	**7.44**
TP-10	109.3299	23.7551	107.7793	23.2875	**1.42**	**1.97**
TP-20	87.4062	20.215	85.8197	19.9767	**1.82**	**1.18**

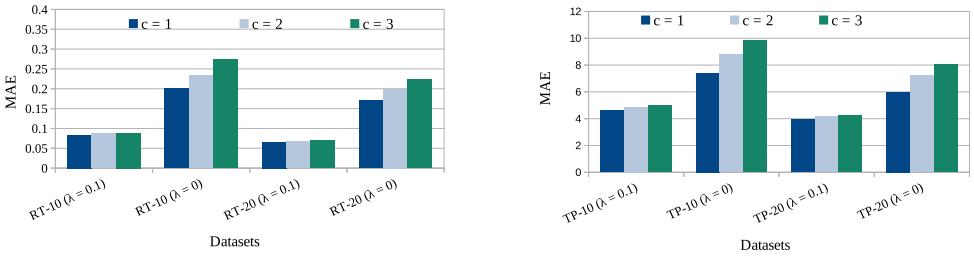
4.2.2.2.3 Impact of Grey-sheep Detection Algorithm. Fig.s 4.7 (a) and (b) show the Grey-sheep detection capability of GRRQP on RT and TP datasets, respectively. We may infer the following observations from Fig.s 4.7 (a) and (b) below:

(i) For $\lambda = 0$, as the value of c decreased and the number of detected GSU and GSS increased and removed, the prediction error decreased. Table 4.8 shows the number of detected GSUs and GSSs for different values of c.

(ii) As we removed 10% of the outliers (i.e., $\lambda = 0.1$) along with the GSU and GSS for every value of c, the performance improvement of GRRQP was quite significant compared to the case with removing the GSU and GSS only without removing any outliers (i.e., $\lambda = 0$). Here, we refer to the comparison between the same colored bars in the two adjacent cases in Fig.s 4.7 (a) and (b), for example, comparing the dark blue bars for RT-10 ($\lambda = 0.1$) with RT-10 ($\lambda = 0$).

(iii) For $\lambda = 0.1$, the decrease in the prediction error with a decrease in the value of c was not that significant compared to the case for $\lambda = 0$. This finding suggests that the presence of a large number of outliers might contribute to the emergence of more grey-sheep instances.

4.2.2.2.4 Performance of GRRQP. Table 4.9 presents the comparison between GRRQP and SORRQP to show the effectiveness of our framework in handling the GSU and GSS. While the SORRQP only addresses sparsity and outliers issues, the GRRQP offers explicit solutions for GSU and GSS in QoS prediction. This experiment was divided into two cases: (i) Case-1: considering all users and GSS ($\mathcal{U}, \mathcal{S}_{\mathcal{G}}$), (ii)



(a) (b)

Figure 4.7: Analysis of grey-sheep detection metric (a) RT and (b) TP datasets

Table 4.10: Comparison of ARRQP with SOTA on grey-sheep removal

Methods	RT-10		RT-20		TP-10		TP-20	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LRMF [77]	0.4785	1.2576	0.4340	1.1422	-	-	-	-
RAP [22]	0.4914	1.4346	0.4531	1.1631	21.4027	-	17.5978	-
HLT [156]	0.5088	-	0.3634	-	28.5370	-	22.8296	-
CAP [16]	0.4997	-	0.4361	-	16.6465	-	14.2685	-
TAP [15]	0.5978	-	-	-	25.7784	-	-	-
ARRQP $\lambda = 0.0$	**0.2343**	**1.0106**	**0.1992**	**0.9450**	**8.7754**	**32.1770**	**7.2216**	**28.7231**
PG (%)	**51.03**	**19.64**	**45.18**	**17.26**	**47.28**	**-**	**49.39**	**-**
ARRQP $\lambda = 0.1$	**0.2079**	**0.8180**	**0.1747**	**0.7475**	**8.1668**	**27.2536**	**6.7368**	**24.4762**
PG (%)	**56.55**	**34.95**	**51.93**	**34.56**	**50.94**	**-**	**52.78**	**-**

* *PG* of ARRQP ($c = 2$) over the second best method highlighted in blue

Case-2: considering GSU and all services $(\mathcal{U}_{\mathcal{G}}, \mathcal{S})$. The improvement of GRRQP over SORRQP for Case-1 and Case-2 are shown in the last two columns of Table 4.9.

4.2.2.2.5 Performance of ARRQP After Grey-sheep Removal.

Table 4.10 demonstrates the performance of ARRQP after removing the data corresponding to GSU and GSS for $c = 2$. Here, we compared our framework with the SOTA methods that attempted to detect GSU and GSS and provide the results after removing them. As observed from Table 4.10, ARRQP outperformed all the methods with a significant improvement margin with or without removing outliers.

Moreover, it is worth mentioning that even though SOTA methods attempted to detect GSU and GSS, they failed to handle them. In contrast to the existing literature, ARRQP detects GSU and GSS and effectively handles them.

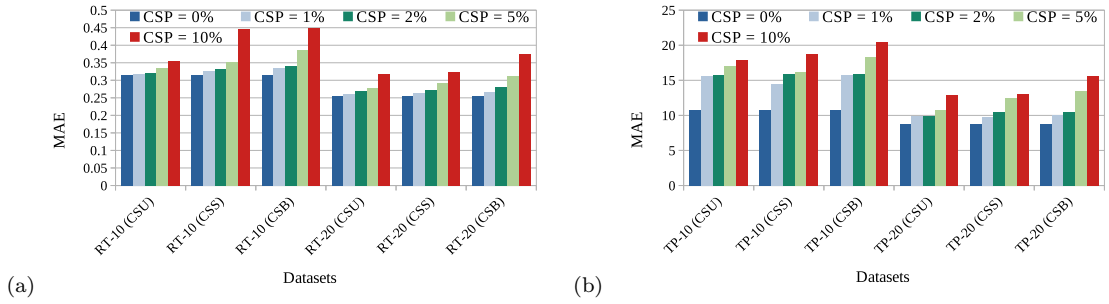


Figure 4.8: Analysis of Cold-start on (a) RT and (b) TP datasets

4.2.2.2.6 Performance of CRRQP. Cold-start scenarios occur when new users or services are added to the system. We divided our experiment into three cases:

(i) CSU: This is the first case where the experiment was designed with a set of cold-start users without any data in the system. A cold-start percentage (CSP) used for the experiment determines the number of cold-start users. The experiment was limited to cold-start users and all the services for this case.

(ii) CSS: This is the second case. Here, the experiment was designed with a set of cold-start services having no data in the system. Here, CSP also determines the number of cold-start services. The experiment was performed on all the users and cold-start services for this case.

(iii) CSB: Final case combines CSU and CSS. In this case, if the value of CSP is considered as x, $x\%$ of the total users and $x\%$ of the total services are designated as the cold-start users and services. The experiment here was performed jointly on all the users with cold-start services and cold-start users with all the services.

Here, we present the analysis of various CSP values. Fig.s 4.8 (a) and (b) show the sensitivity of cold-start on the performance of ARRQP on RT and TP datasets, respectively. As evident from Fig.s 4.8 (a) and (b), the performance of ARRQP declined as the value of CSP increased. In addition to the lack of information on cold-start users and services, the performance degradation of ARRQP was also attributed to the increasing sparsity caused by the rising CSP value.

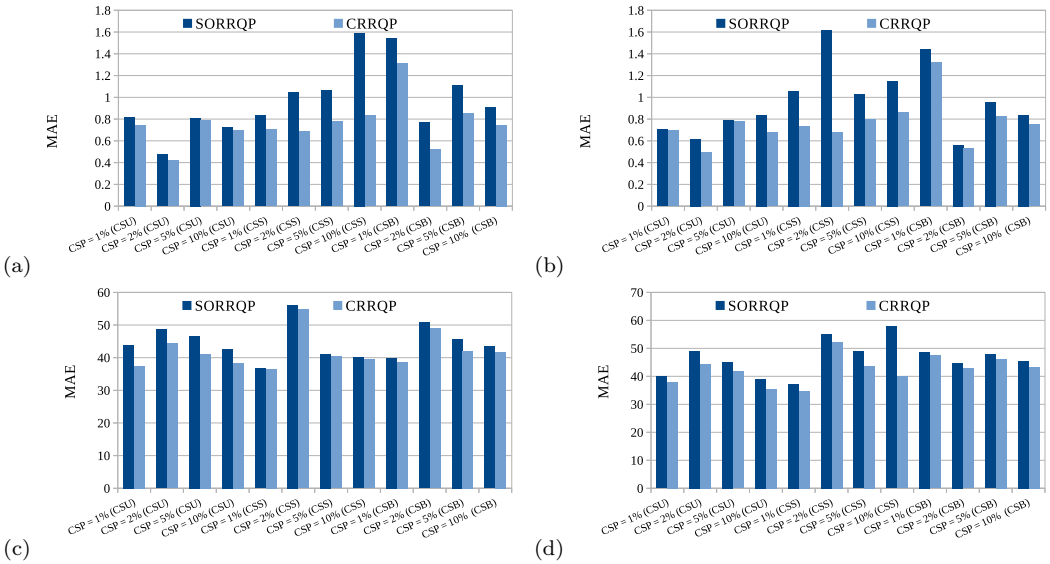


Figure 4.9: Performance of CRRQP on (a) RT-10, (b) RT-20, (c) TP-10, (d) TP-20 datasets

Table 4.11: Comparison of CRRQP with SOTA under cold-start for CSB on RT-10.

Methods	MAE			RMSE		
	0%	5%	10%	0%	5%	10%
GMEAN	1.0154	1.0300	1.0100	1.9678	1.9680	1.9702
USMF [38]	0.5499	0.5800	0.6100	1.4148	1.4954	1.5800
GeoMF1 [12]	0.4890	0.5200	0.5500	1.2022	1.3100	1.4000
GeoMF2 [12]	0.4610	0.5100	0.5200	1.1774	1.2900	1.3800
CRRQP	**0.3137**	**0.3874**	**0.4479**	**1.1456**	**1.2435**	**1.2905**
PG (%)	**46.95**	**31.64**	**16.09**	**2.77**	**3.73**	**6.93**

GMEAN: uses the average QoS value to make predictions

Fig.s 4.9 (a)–(d) showcase the comparative performance of CRRQP and SORRQP on RT-10, RT-20, TP-10, and TP-20 datasets, respectively. As evident from Fig.s 4.9(a)–(d), CRRQP consistently outperformed SORRQP, exhibiting a notable average improvement of 18.94% and 7.04% on RT and TP datasets, respectively.

Table 4.11 illustrates the performance of CRRQP compared to SOTA methods for CSB with CSP of 0%, 5%, and 10% on the RT-10 dataset. Table 4.11 demonstrates the significant performance enhancement achieved by CRRQP over the existing SOTA methods that addressed the cold-start problem.

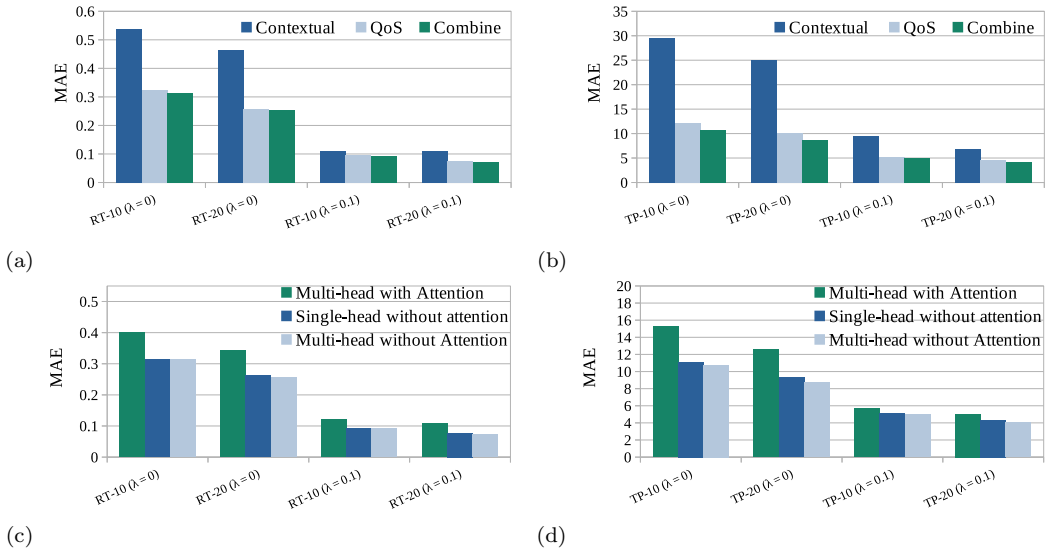


Figure 4.10: Feature ablation study on (a) RT, (b) TP; Model ablation study on (c) RT, (d) TP datasets

4.2.2.3 Ablation Study

4.2.2.3.1 Feature Ablation Study. Fig.s 4.10 (a) and (b) illustrate the influence of different features used in our framework. From Fig.s 4.10 (a) and (b), we can infer the following.

(i) ARRQP with contextual features exhibited the worst performance. Although QoS features had a greater impact than contextual features, ARRQP combining both contextual and QoS features outperformed the same with only QoS features by 2.01% and 12.63% on RT and TP datasets, respectively, on average.

(ii) For $\lambda = 0.1$, all three models showed significant performance improvements compared to the case for $\lambda = 0$. The improvement percentages are $78.07\%, 70.02\%, 70.93\%$ on RT, and $70.21\%, 55.75\%, 53.49\%$ on TP datasets for ARRQP with contextual features, QoS features and combined features, respectively.

4.2.2.3.2 Model Ablation Study. We studied the individual performance of GRRQP and CRRQP as compared to SORRQP in Table 4.9 and Fig.s 4.9 (a)–(d), respectively. These results emphasize the importance of GRRQP handling grey-sheep

users and services and CRRQP handling cold-start situations alongside SORRQP handling sparsity and outliers.

4.2.2.3.3 Comparative Model Study Here, we present a comparative study between three models: the single-head without attention model, the multi-head without attention model (ARRQP), and the multi-head with attention model. Fig.s 4.10 (c) and (d) present the performance of all three models on RT and TP datasets, respectively. It is evident from Fig.s 4.10 (c) and (d) that ARRQP outperformed the multi-head with attention model [68] by an average improvement of 23.69% and 30.08% on RT and TP datasets, respectively. Additionally, it also surpassed the single-head without attention model by an average improvement of 1.34% and 4.74% on RT and TP datasets, respectively. A similar trend is observed after 10% outliers removal.

4.2.2.4 Impact of Hyper-parameters

4.2.2.4.1 Impact of γ. The hyper-parameter γ is present in the Cauchy loss function. Fig.s 4.11 (a) and (b) illustrate the performance of ARRQP with the change in the value of γ on RT and TP datasets, respectively. Among different γ values, we observed that $\gamma = 0.25$ and $\gamma = 10$ achieved the best performance for all training densities on the RT and TP datasets, respectively. The same trend was followed after removing the 10% of outliers from the datasets.

4.2.2.4.2 Impact of Number of Heads (N_h). We tuned our model with different numbers of heads. The experimental results are shown in Fig.s 4.11 (c) and (d) for the RT and TP datasets, respectively. Here, we have the following observations:

(i) For RT datasets, although there is no fixed pattern, one head is sufficient to achieve the best performance at 10% training density (TrD), while three heads performed best at 20% TrD. Moreover, removing 10% outliers enhances the model performance.

(ii) For TP datasets, the best performance is observed with four heads at 10% TrD and five heads at 20% TrD.

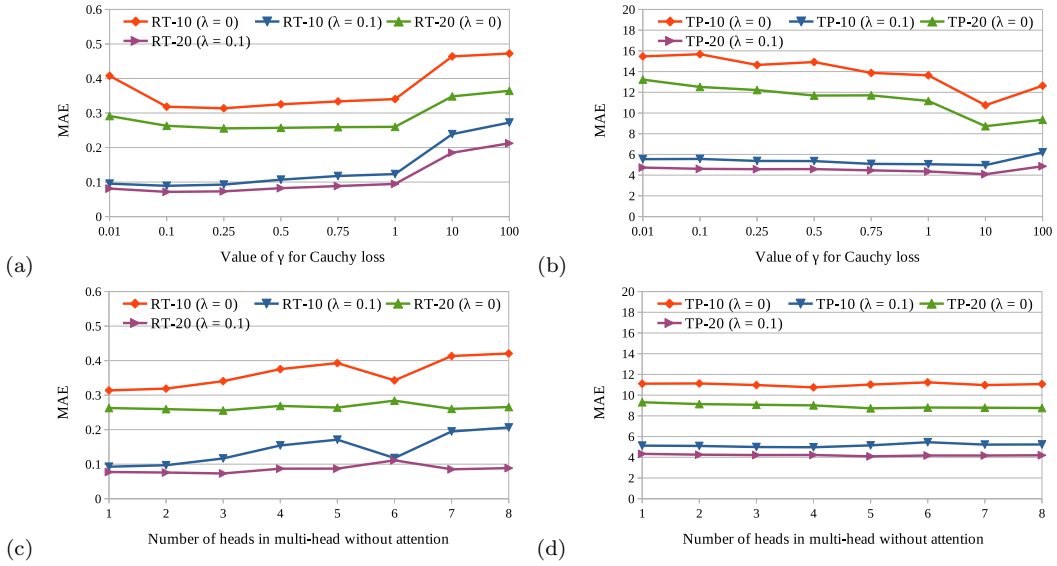


Figure 4.11: Impact of γ on (a) RT and (b) TP datasets. Impact of the number of heads (N_h) in the multi-head model without attention on (c) RT and (d) TP datasets.

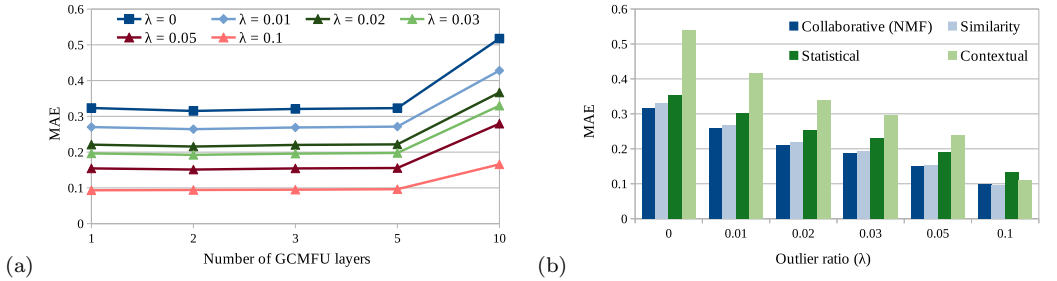


Figure 4.12: (a) Impact of number of MhGCMFU (t), and (b) Contribution of different QoS features on RT-10 dataset

4.2.2.4.3 Impact of Number of MhGCMFU Layers (t). We analyze the impact of the number of MhGCMFU layers (denoted as t) on ARRQP's performance. As shown in Fig. 4.12 (a), the lowest prediction error was achieved when $t = 2$, and the variation in error remained minimal for $t \leqslant 5$. While deeper GCNs typically suffer from the over-smoothing problem [73], ARRQP mitigated this issue through channel-wise concatenation of outputs from each MhGCMFU layer. However, when t exceeded 5, prediction accuracy deteriorated significantly due to increased over-smoothing. This underscores the importance of carefully selecting t to ensure robust and credible performance of ARRQP.

Table 4.12: Confidence Intervals

CL	RT-10	RT-20	TP-10	TP-20
90%	(0.3052, 0.3055)	(0.2399, 0.2402)	(10.4042, 10.4152)	(8.3365, 8.3450)
95%	(0.3051, 0.3055)	(0.2398, 0.2402)	(10.4032, 10.4162)	(8.3357, 8.3458)
99%	(0.3051, 0.3056)	(0.2398, 0.2402)	(10.4011, 10.4183)	(8.3341, 8.3474)
MAE	0.3053	0.2400	10.4097	8.3408
Std Dev	0.0076	0.0070	0.2672	0.2076

4.2.2.4.4 Impact of Various Features in Initial Embedding. We analyze the impact of various QoS and contextual feature types on the performance of ARRQP. Building upon our earlier ablation study, which demonstrated that the combination of QoS and contextual features yields better performance than using them individually, we further evaluate the standalone effectiveness of statistical, collaborative (NMF-based), similarity, and contextual features. As shown in Fig. 4.12 (b), collaborative features contributed the most when used independently. Nevertheless, the optimal performance of ARRQP was achieved when all feature types are combined, underscoring the complementary nature of these diverse representations.

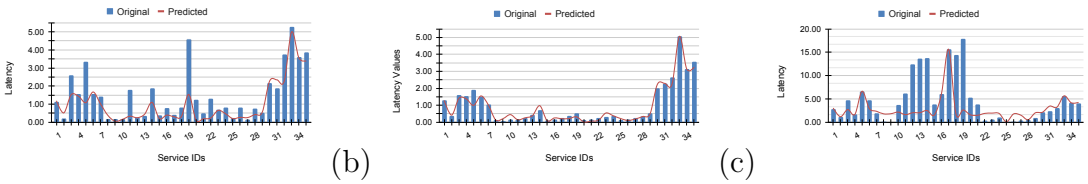
4.2.2.5 Statistical Significance Testing

To establish the reliability of ARRQP, we assess the statistical significance test via confidence intervals (CIs) [157]. A CI essentially provides a range around a measurement, indicating the precision of that measurement. Typically, CIs are computed for various confidence levels (CL), such as $90\%, 95\%$, and 99%. A $x\%$ CL implies that there is a $(100 - x)\%$ chance of uncertainty in the experiments of being wrong.

The insights from Table 4.12 enable us to evaluate the precision of ARRQP under diverse conditions, facilitating well-informed decisions about its reliability. Notably, as we move from lower to higher CL, the CI tends to become wider.

4.2.3 A Use-Case for ARRQP

This case study highlights the importance and effectiveness of ARRQP in delivering user-specific low-latency service recommendations. We expose ARRQP in smart



(a) (b) (c)

Figure 4.13: ARRQP prediction analysis over latency for three users with 35 service invocations: (a) u_5, (b) u_{25}, and (c) u_{50}

transportation domain to obtain reliable AV services. Fig.s 4.13 (a)-(c), analyze 35 AV services from the gRPC dataset, illustrated in sec. 4.2.1.2, we observe that no single service consistently offers the lowest latency across all users—for example, service s_{27} is optimal for user u_5, while s_9 is best for users u_{25} and u_{50}. Moreover, a service optimal for one user may perform poorly for another; s_{27} provides u_5 a latency of 14.86 ms but yields a much higher 33.66 ms for u_{25}, who instead gets just 7.03 ms from s_9. ARRQP efficiently anticipates such variations and recommends s_{25} for u_{25}, achieving a latency of 10.53 ms—close to the user's optimal. These insights underscore ARRQP's ability to guide latency-aware recommendations in safety-critical scenarios like autonomous driving.

4.3 Summary

This chapter presents an anomaly-resilient real-time QoS prediction framework (ARRQP), designed to achieve highly accurate QoS prediction in negligible time by addressing various challenges, including data sparsity, the presence of outliers, grey-sheep instances, and the cold-start problem. ARRQP proposes a multi-head graph convolution matrix factorization model to capture complex relationships and dependencies among users and services. By doing so, it enhances its ability to accurately predict QoS, even in the face of data limitations. Furthermore, ARRQP integrates contextual information and collaborative insights, enabling a holistic understanding of user-service interactions. Robust loss functions are employed to mitigate the impact of outliers during model training, improving predictive accuracy. Additionally, ARRQP introduces a sparsity-resilient method for detecting grey-sheep users or services. These

distinctive instances are subsequently handled separately for QoS prediction, ensuring tailored and accurate predictions. Moreover, the cold-start problem is addressed as a distinct challenge, emphasizing the importance of contextual features. ARRQP also exhibited a high responsiveness and scalability, rendering it well-suited for integration into real-time systems. This characteristic positions ARRQP as a valuable tool for applications where timely and efficient QoS prediction is essential.

4.4 Limitations

While ARRQP advances reliable and real-time QoS prediction under noisy and anomalous conditions, several limitations remain:

(i) ARRQP relies on predefined anomaly detection and mitigation mechanisms. Incorporating more advanced, adaptive, or learning-based anomaly modeling strategies could further enhance prediction robustness and accuracy.

(ii) ARRQP does not explicitly model temporal dependencies in QoS evolution. Developing a time-aware extension is essential for capturing dynamic user–service interaction patterns and improving prediction performance in evolving service environments.

(iii) ARRQP models user–service relationships using pairwise graph interactions. However, it does not explicitly capture higher-order collaborative dependencies involving multiple users, services, and contextual factors simultaneously, which may limit its representational capability under highly sparse conditions.

(iv) The anomaly handling and graph modeling components operate independently, without a fully integrated credibility-aware representation learning mechanism, which may reduce overall robustness in highly noisy environments.

Chapter 5

Anomaly Resilient Temporal QoS Prediction using Hypergraph Convoluted Transformer Network

We identify three key factors causing accuracy degradation in temporal QoS prediction: (a) *Data Deficiency*: Sparse QoS experiences, especially in cold-start scenarios, arise from limited user-service interactions [71]. (b) *Data Credibility*: Dynamic environments introduce outliers, while the grey-sheep problem complicates predictions for atypical instances with unique traits. (c) *Data Learning*: Traditional methods using similarity features [6,7] or linear matrix decomposition [32,37] fail to capture complex user-service relationships over time. While some approaches leverage GCNs [55, 71] to enhance feature learning, they struggle with temporal dynamics, and models like ARIMA [60], RNNs [18,72], and Transformers [158] enhance temporal feature learning, they often overlook local spatial context and struggle with data anomalies, impacting accuracy. Our previous study, TPMCF [159], employs distinct architectures to sepa-

This chapter is derived from:

- **Suraj Kumar**, Soumi Chattopadhyay and Chandranath Adak, "Anomaly-Resilient Temporal QoS Prediction Using Hypergraph Convoluted Transformer Network," in IEEE Transactions on Network and Service Management, vol. 23, pp. 3556-3568, 2026, doi: 10.1109/TNSM.2026.3674650.

rately capture spatial and temporal features but faces challenges in managing certain data anomalies, which impacts its prediction accuracy.

Considering the abovementioned extremity, we propose an anomaly-resilient temporal QoS prediction framework called Hypergraph Convoluted Transformer Network (HCTN), an end-to-end architecture with five key components: (a) The first module applies non-negative matrix decomposition to extract latent user and service features, addressing data sparsity and cold-start issues, (b) The second module uses hypergraph collaborative filtering to capture higher-order user-service features through hypergraph convolution over hyperedges, further addressing sparsity, (c) To tackle the grey-sheep problem, the third module emphasizes local characteristics of grey-sheep users and services, enhancing the effectiveness of collaborative filtering, (d) The next module refines temporal granularity by feeding updated embeddings into an enhanced transformer network that combines multi-head attention [130] with parallel 1D-convolutional layers and fully connected networks. This enables the model to capture fine- and coarse-grained temporal patterns, providing a rich representation of both short-term variations and long-term trends, and (e) The final module predicts QoS by integrating collaborative, spatial, and temporal features, with the entire model trained end-to-end using an outlier-resilient loss function to handle anomalies. This framework effectively addresses data sparsity, cold-start, grey-sheep, and temporal dynamics challenges, providing accurate QoS predictions. In summary, this chapter has following contributions:

(i) **Novelty in model architecture**: We introduce the Hypergraph Convoluted Transformer Network (HCTN) for temporal QoS prediction. HCTN combines hypergraph learning for higher-order collaborative filtering with dual-channel Transformer networks, capturing dynamic features from fine-to-coarse-grained levels and using multi-head attention to manage various contexts.

(ii) **Handling data deficiency**: We address data sparsity using higher-order collaborative filtering with diverse hyperedges in second-order user-service graphs. Additionally, non-negative matrix decomposition uncovers hidden user and service

preferences, improving cold-start performance and enhancing collaborative filtering features.

(iii) **Tackling data credibility**: To address data credibility, we apply two sparsity-resilient unsupervised algorithms to detect outliers and grey-sheep. Outliers are handled with a logarithmic penalty on training errors, while grey-sheep are managed by prioritizing their unique patterns over global collaborative filtering features, improving QoS prediction.

(iv) **Rich data representation**: In our framework, domain knowledge features, such as latent user preferences and service characteristics, are combined with higher-order complex spatio-temporal features to effectively capture the intricate triadic relationships among users, services, and time, thereby enhancing overall data representation.

(v) **Extensive experiments**: We conducted comprehensive experiments using the QoS benchmark dataset, WSDREAM-2 [127], for two different QoS parameters, response time and throughput. These experiments highlight the superior performance of HCTN compared to contemporary methods. Additionally, our ablation study validates the effectiveness of each module within the proposed framework.

The remaining chapter is organized as follows: Section 5.1 details our framework. Section 5.2 presents the experiments and comparative studies. Finally, Section 5.3 concludes this chapter.

5.1 Proposed Method

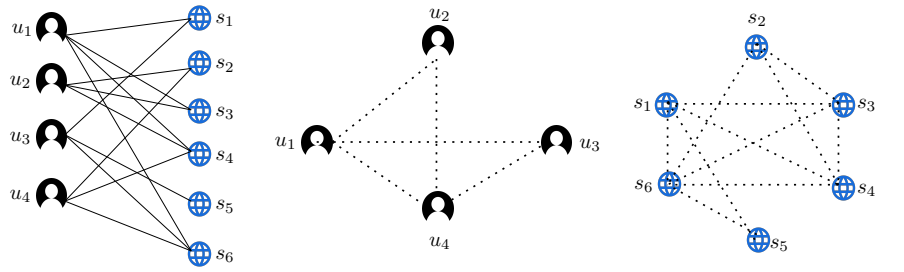
The classical temporal QoS prediction problem (Def. 5) is challenged by three fundamental issues: data sparsity, data credibility, and insufficient representation learning. To address these limitations, this chapter presents an anomaly-resilient temporal QoS prediction framework that systematically tackles six critical research questions: (i) limited user–service interactions, (ii) cold-start scenarios, (iii) outlier contamination,

(iv) grey-sheep behavior, and the integration of (v) domain-specific features and (vi) higher-order structural representations.

This section introduces our proposed framework for anomaly-resilient real-time temporal QoS prediction comprising an end-to-end Hypergraph Convoluted Transformer Network (HCTN) that is composed of five primary modules: *(i) Global Pattern Adaptation Module (GPAM)*, *(ii) Hypergraph Collaborative Filtering Module (HCFM)*, *(iii) Grey-sheep Mitigation Module (GMM)* with two sub-modules: the Grey-sheep Detection Module (GDM), and the Local Pattern Adaptation Module (LPAM), *(iv) Temporal Granularity Extraction Module (TGEM)*, and *(v) Comprehensive QoS Prediction Module (CQPM)*. By leveraging the diverse features obtained from various modules of HCTN, the overall goal of the framework is to deliver highly accurate real-time temporal QoS predictions while effectively addressing the aforementioned challenges. A brief overview of HCTN is presented in Fig. 5.2. We now illustrate each module of HCTN in detail.

5.1.1 Global Pattern Adaptation Module (GPAM)

The primary purpose of the GPAM is to provide domain-specific features by preserving the global characteristics of user-service interactions at a given time-step t. These features contribute to generating initial feature embeddings for each user and service. GPAM employs Non-negative Matrix Decomposition (NMD) [32] to uncover hidden QoS invocation patterns among users and services.



(a) First-order user-service interaction graph (FIG) (b) Second-order user interaction graph (SUIG) (c) Second-order service interaction graph (SSIG)

Figure 5.1: Decomposition of QIHG: (a) FIG, (b) SUIG, and (c) SSIG

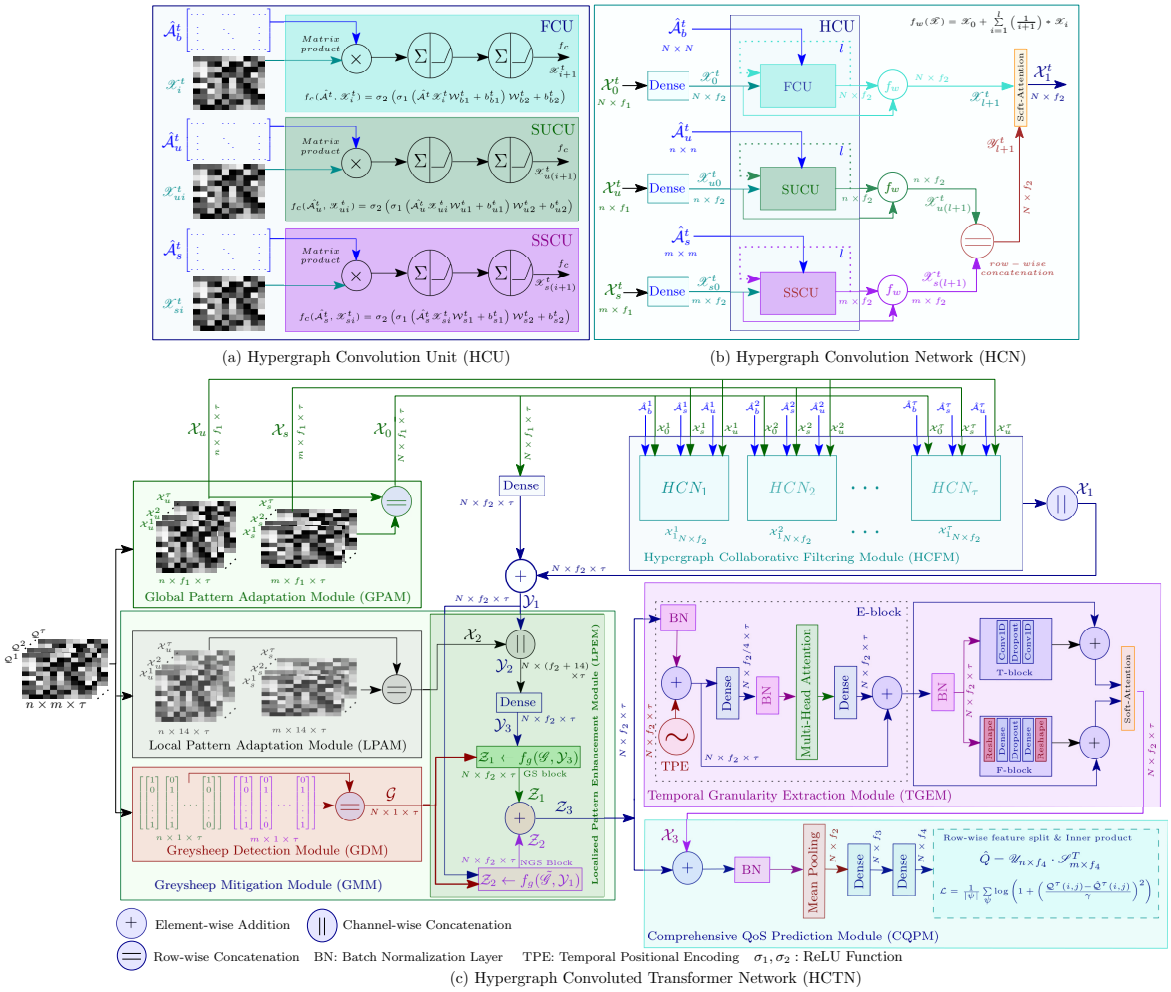


Figure 5.2: Hypergraph Convoluted Transformer Network (HCTN) Framework

Given a QoS invocation matrix $\mathcal{Q}^t$ for n users and m services, NMD decomposes $\mathcal{Q}^t$ into two low-rank latent features matrices, $\mathcal{X}_u^t$ and $\mathcal{X}_s^t$, such that:

$$\mathcal{Q}^t_{n\times m}(i,j) \approx \left[(\mathcal{X}_u^t)_{n\times f_1} \cdot (\mathcal{X}_s^t)^T_{m\times f_1}\right](i,j), \quad \forall \mathcal{Q}^t(i,j) \geqslant 0 \tag{5.1}$$

By applying NMD to each of the previous $\mathcal{T}$ time-steps, GPAM extracts f_1-dimensional features for users and services, as depicted in Fig. 5.2(c). This process yields two feature tensors: $\mathcal{X}_u$ with dimensions $n \times f_1 \times \mathcal{T}$, and $\mathcal{X}_s$ with dimensions $m \times f_1 \times \mathcal{T}$, corresponding to users and services, respectively. These user and service features are further combined into a user-service feature tensor $\mathcal{X}_0$ with dimensions $N \times f_1 \times \mathcal{T}$, where $N = n + m$, by row-wise concatenating each $\mathcal{X}_u^t$ and $\mathcal{X}_s^t$. These features are utilized in Section 5.1.2. Notably, GPAM produces dense feature tensors,

which help mitigate the cold-start problem without requiring additional interventions.

5.1.2 Hypergraph Collaborative Filtering Module (HCFM)

GPAM effectively provides collaborative features that partially address the data-sparsity problem. However, it falls short in capturing higher-order, non-linear, and complex features of users and services that are essential for high prediction performance. To address this, various GCN-based frameworks [55, 71, 108, 159] have been introduced in the literature, utilizing bipartite graph representations of user-service QoS interactions. However, user-service interactions can extend beyond bipartite connections, leading to multidimensional relationships represented by hyper-edges. In this chapter, we introduce a Hypergraph Convolution Network (HCN), as presented in Fig. 5.2(b), to exploit these connections. HCN primarily leverages graph convolution over various graph representations constructed using different hyper-edges, as shown in Fig. 5.1. Before discussing the details of this module, we first introduce the concept of a hypergraph in the context of QoS invocation.

5.1.2.1 Hypergraph to Represent QoS Invocation

The connectivity between users and services to form a graph structure can be established through various methods, such as by shared autonomous systems or geographical locations. However, due to privacy concerns, we may not always have access to such details. To address this issue, we construct a hypergraph structure based on their past interactions recorded in the QoS invocation matrix at a given time-step t. For each time-step t, we define the QoS invocation hypergraph in Definition 22.

Definition 22. *[**QoS Invocation Hypergraph (QIHG)**]: Given a set of users $\mathcal{U}$ and services $\mathcal{S}$, QIHG at time-step t, denoted by $\mathcal{G}^t$, is defined as $\mathcal{G}^t = (\mathcal{V}_u \cup \mathcal{V}_s, \mathcal{E}^t)$. $\mathcal{V}_u$ represents the set of users $\mathcal{U}$, and $\mathcal{V}_s$ represents the set of services $\mathcal{S}$. The hyperedges $\mathcal{E}^t$ extend beyond simple pairwise connections and are of two types: (i) a hyperedge $\{v_i^u, v_j^s, v_k^u\}$, indicating that both users u_i and u_k corresponding to v_i^u and $v_k^u \in \mathcal{V}_u$ invoked service s_j corresponding to $v_j^s \in \mathcal{V}_s$ at time-step t; and (ii) a hyperedge*

$\{v_i^s, v_j^u, v_k^s\}$, *indicating that both services* s_i *and* s_k *corresponding to* v_i^s *and* $v_k^s \in \mathcal{V}_s$ *were invoked by user* u_j *corresponding to* $v_j^u \in \mathcal{V}_u$ *at time-step* t. ■

We simplify the hypergraph $\mathcal{G}^t$ by decomposing it into three distinct graphs (shown in Fig. 5.1), each capturing a specific type of relationship among users and services:

5.1.2.1.1 First-order user-service Interaction Graph (FIG). The FIG, denoted as $\mathcal{G}_f^t = (\mathcal{V}_u \cup \mathcal{V}_s, \mathcal{E}_f^t)$, captures direct interactions between users and services. An edge $(v_i^u, v_j^s) \in \mathcal{E}_f^t$ exists if the user corresponding to $v_i^u \in \mathcal{V}_u$ invoked the service corresponding to $v_j^s \in \mathcal{V}_s$ at time-step t. $\mathcal{G}_f^t$ is represented by an adjacency matrix $\mathcal{A}^t \in \{0,1\}_{N\times N}$, where $N = n + m$.

5.1.2.1.2 Second-order User Interaction Graph (SUIG). The SUIG, denoted by $\mathcal{G}_u^t = (\mathcal{V}_u, \mathcal{E}_u^t)$, represents the relationship between two users. An edge $(v_i^u, v_j^u) \in \mathcal{E}_u^t$ exists if the users corresponding to v_i^u and v_j^u both invoked at least one common service at t, i.e., $\exists s_k \in \mathcal{S}$, such that $\mathcal{Q}(i,k,t) \neq 0$ and $\mathcal{Q}(j,k,t) \neq 0$, which means, $\exists v_k^s \in \mathcal{V}_s$, such that $\{v_i^u, v_k^s, v_j^u\} \in \mathcal{E}^t$.

5.1.2.1.3 Second-order Service Interaction Graph (SSIG). The SSIG, denoted by $\mathcal{G}_s^t = (\mathcal{V}_s, \mathcal{E}_s^t)$, represents the relationship between two services. An edge $(v_i^s, v_j^s) \in \mathcal{E}_s^t$ exists if the services corresponding to v_i^s and v_j^s both were invoked by at least one common user at t. This means $\exists u_k \in \mathcal{U}$ such that $\mathcal{Q}(k,i,t) \neq 0$ and $\mathcal{Q}(k,j,t) \neq 0$, indicating that $\{v_i^s, v_k^u, v_j^s\} \in \mathcal{E}^t$ for some $\exists v_k^u \in \mathcal{V}_u$.

5.1.2.2 Hypergraph Convolution Units (HCU)

We now explain the process of extracting higher-order collaborative features using hypergraph convolution on various graphs, including $\mathcal{G}_f^t$, $\mathcal{G}_u^t$, and $\mathcal{G}_s^t$, each representing distinct relationships. Fig. 5.2(a) provides an overview of the Hypergraph Convolutional Unit (HCU) architecture. The HCU consists of three convolution units: FCU, SUCU, and SSCU, as described below.

5.1.2.2.1 First-order Convolution Unit (FCU). At each time-step t, the FCU operates on the FIG $\mathcal{G}_f^t$. It takes two inputs: the normalized adjacency matrix $\hat{\mathcal{A}}^t$ of $\mathcal{G}_f^t$ and a pre-processed feature embedding $\mathscr{X}_i^t$ representing all users and services. The FCU then applies graph convolution [67] function f_c to these inputs to extract nonlinear collaborative features, as outlined in the equation presented in Fig. 5.2(a). The normalized adjacency matrix $\hat{\mathcal{A}}^t = (\mathcal{D}^t)^{-1/2} \cdot (\mathcal{A}^t + \mathbb{I}) \cdot (\mathcal{D}^t)^{-1/2}$ is obtained using symmetric normalization. The diagonal degree matrix, derived from $\mathcal{A}^t$, and defined as $\mathcal{D}^t(i,i) = \sum_j \mathcal{A}^t(i,j)$, is used to calculate $\hat{\mathcal{A}}^t$. It may be noted that using $\hat{\mathcal{A}}^t$ helps mitigate numerical instability and sensitivity issues caused by high-degree nodes, unlike when using $\mathcal{A}^t$ directly. Additionally, the identity matrix ($\mathbb{I}$) is added during normalization to include self-features in the convolution process.

5.1.2.2.2 Second-order User Convolution Unit (SUCU) and Second-order Service Convolution Unit (SSCU). At each time-step t, the SUCU and SSCU operate on $\mathcal{G}_u^t$ and $\mathcal{G}_s^t$, respectively, applying graph convolution functions $f_c(\hat{\mathcal{A}}_u^t,\ \mathscr{X}_{ui}^t)$ and $f_c(\hat{\mathcal{A}}_s^t,\ \mathscr{X}_{si}^t)$ to independently extract nonlinear collaborative features $\mathscr{X}_{u(i+1)}^t$ for users and $\mathscr{X}_{s(i+1)}^t$ for services, as shown in the equations in Fig. 5.2(a).

Here, $\hat{\mathcal{A}}_u^t$ and $\hat{\mathcal{A}}_s^t$ represent the normalized matrices that the SUCU and SSCU receive as input. However, unlike the FCU, which uses adjacency matrices derived from $\mathcal{G}_f^t$, we instead leverage the incidence matrix $\mathcal{H}^t$, which is also derived from $\mathcal{G}_f^t$. This matrix is defined as:

$$\mathcal{H}^t = \begin{cases} h_{ij}^t = 1, & \text{if } \mathcal{Q}(i,j,t) \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

The derivation to compute $\hat{\mathcal{A}}_u^t$ and $\hat{\mathcal{A}}_s^t$ is presented in Eq.s 5.3 and Eq. 5.4:

$$\hat{\mathcal{A}}_u^t = \left(\mathcal{D}_u^t\right)^{-1/2} \cdot \mathcal{H}^t \cdot \left(\mathcal{D}_s^t\right)^{-1} \cdot \left(\mathcal{H}^t\right)^T \cdot \left(\mathcal{D}_u^t\right)^{-1/2} \tag{5.3}$$

$$\hat{\mathcal{A}}_s^t = \left(\mathcal{D}_s^t\right)^{-1/2} \cdot \left(\mathcal{H}^t\right)^T \cdot \left(\mathcal{D}_u^t\right)^{-1} \cdot \mathcal{H}^t \cdot \left(\mathcal{D}_s^t\right)^{-1/2} \tag{5.4}$$

where, the degree matrices $(\mathcal{D}_u^t)_{n\times n}$ and $(\mathcal{D}_s^t)_{m\times m}$ are derived from the adjacency matrices $\mathcal{A}_u^t$ and $\mathcal{A}_s^t$, corresponding to $\mathcal{G}_u^t$ and $\mathcal{G}_s^t$, respectively. These degree matrices are computed in a similar manner to $\mathcal{D}^t$.

We now introduce the Hypergraph Convolution Network (HCN), where the HCU serves as the primary component.

5.1.2.3 Hypergraph Convolution Network (HCN)

Fig. 5.2(b) illustrates an overview of the HCN. At each time-step t, HCN receives feature embeddings $\mathcal{X}_0^t$, $\mathcal{X}_u^t$, and $\mathcal{X}_s^t$ as inputs, which are generated by the GPAM module. These embeddings are first processed through a dense layer, yielding $\mathscr{X}_0^t$, $\mathscr{X}_{u0}^t$, and $\mathscr{X}_{s0}^t$, which are then fed into HCU in a loop l times.

To counteract potential information loss caused by the over-smoothing issue commonly encountered with multiple graph convolution layers [160], we aggregate the outputs of each layer using the function

$$f_w(\bar{\mathscr{X}}) = \mathscr{X}_0 + \sum_{i=1}^{l} (1/(i+1))\, \mathscr{X}_i \tag{5.5}$$

where, $\bar{\mathscr{X}} = (\mathscr{X}_0, \mathscr{X}_1, \ldots, \mathscr{X}_l)$ is a tuple of the convolution outputs of length $(l+1)$. This method balances complexity and feature retention by assigning progressively smaller weights to deeper layer outputs, preserving critical information. Finally, we compute $f_w(\bar{\mathscr{X}}^t)$, $f_w(\bar{\mathscr{X}}_u^t)$, and $f_w(\bar{\mathscr{X}}_s^t)$ to generate $\mathscr{X}_{l+1}^t$, $\mathscr{X}_{u(l+1)}^t$, and $\mathscr{X}_{s(l+1)}^t$, respectively.

We then aggregate $\mathscr{X}_{u(l+1)}^t$ and $\mathscr{X}_{s(l+1)}^t$ through row-wise concatenation to form $\mathscr{Y}_{l+1}^t$. This concatenated matrix is then combined with $\mathscr{X}_{l+1}^t$ using soft-attention [161] to generate $\mathcal{X}_1^t$, which represents the refined, enriched collaborative features as the output of the HCN block.

5.1.2.4 Hypergraph Collaborative Filtering

We now discuss the working of the HCFM. The HCFM leverages the HCN across the past $\mathcal{T}$ time-steps. For each HCN_t where $1 \leqslant t \leqslant \mathcal{T}$, the inputs include the normalized matrices $\hat{\mathcal{A}}^t$, $\hat{\mathcal{A}}_u^t$, and $\hat{\mathcal{A}}_s^t$, along with the feature embeddings $\mathcal{X}_0^t$, $\mathcal{X}_u^t$, and $\mathcal{X}_s^t$. The outputs from each HCN block are concatenated channel-wise to produce the final embedding $\mathcal{X}_1$ of size $N \times f_2 \times \mathcal{T}$.

It is worth noting that the HCFM leverages higher-order, nonlinear, and complex collaborative features, which help alleviate data sparsity issues by utilizing the additional interactions captured through $\mathcal{G}_u^t$ and $\mathcal{G}_s^t$, for each t, where $1 \leqslant t \leqslant \mathcal{T}$. However, during the generation of these collaborative features, crucial initial features that are beneficial for cold-start scenarios may not be preserved. To address this, we introduce a skip connection from the GPAM, where $\mathcal{X}_0$ is resized and added to $\mathcal{X}_1$, producing the feature tensor $\mathcal{Y}_1$ of the same size. This skip connection mitigates the cold-start issue and provides stability in training the HCTN. In the next section, we incorporate the module designed to tackle the grey-sheep problem.

5.1.3 Grey-sheep Mitigation Module (GMM)

While exploring collaborative features, GPAM and HCFM assume user and service preferences are consistent and QoS data remain reliable with time. However, a few user or service characteristics may not be coherent with others due to their deviant pattern of QoS invocations, referred to as Grey-sheep [86]. Finding such users (services) and selectively fueling them with their own features obtained from their QoS profiles is essential for robust QoS prediction. In this section, we specifically deal with the grey-sheep instances whose presence suffers collaborative filtering methods and degrades QoS prediction performance. This module consists of three key components: (a) *Grey-sheep Detection Module (GDM)*: Identifies grey-sheep users (GSU) and grey-sheep services (GSS) within the given QoS data, (b) *Local Pattern Adaptation Module (LPAM)*: Extracts local features for users and services, and (c) *Localized Pattern Enhancement Module (LPEM)*: Injects these local features to address the specific challenges posed by grey-sheep instances.

5.1.3.1 Grey-sheep Detection Module (GDM)

The primary objective of the GDM is to identify grey-sheep instances within the dataset. These instances refer to GSU or GSS whose preferences deviate significantly from the norm, resulting in lower QoS prediction performance. Inspired by the ap-

proach in [86] for detecting atypical instances, the GDM, as shown in Fig. 5.2(c), is specifically designed to identify these grey-sheep instances. Given the QoS invocation tensor $\mathcal{Q}_{n\times m\times \mathcal{T}}$ for the past $\mathcal{T}$ time-steps, the GDM employs a metric called the Grey-sheep Discrepancy Index (GDI) to assess each user and service for potential grey-sheep characteristics.

5.1.3.1.1 Grey-sheep Discrepancy Index (GDI). We adopt the concept introduced in [86] to define the GDI. The GDI of a user u_i (denoted by $\mathbb{G}^t(u_i)$) or a service s_j (denoted by $\mathbb{G}^t(s_j)$) is a scalar value derived by comparing the QoS invocation profile of u_i (i.e., $\mathcal{Q}^t(i,.)$) or of s_j (i.e., $\mathcal{Q}^t(.,j)$) against the overall QoS mean for u_i (represented by $\mu^t(u_i) = mean(\mathcal{Q}^t(i,.))$) or s_j (represented by $\mu^t(s_j) = mean(\mathcal{Q}^t(.,j))$). This comparison also takes into account the individual service or user means when u_i interacts with a specific service or when s_j is invoked by a particular user.

The primary goal of introducing GDI is to determine how a user or service may exhibit unique characteristics that deviate from the general norms of other users and services. The GDI for a user u_i (denoted as $\mathbb{G}^t(u_i)$) or a service s_j (denoted as $\mathbb{G}^t(s_j)$) at time-step t is a scalar value calculated by comparing the QoS invocation profile of u_i (i.e., $\mathcal{Q}^t(i,.)$) or s_j (i.e., $\mathcal{Q}^t(.,j)$) with the overall QoS mean for u_i (denoted as $\mu^t(u_i) = mean(\mathcal{Q}^t(i,.))$) or s_j (denoted as $\mu^t(s_j) = mean(\mathcal{Q}^t(.,j))$). This comparison incorporates individual means for each service when u_i interacts with a service, or for each user when s_j is invoked by a user. The mathematical definitions of $\mathbb{G}^t(u_i)$ and $\mathbb{G}^t(s_j)$ are provided in Eq.s 5.6 and 5.7, respectively.

$$\mathbb{G}^t(u_i) = \frac{\sum\limits_{s_j \in \mathcal{S}_i^t} \Big(|\mathcal{Q}^t(i,j) - \mu^t(u_i) - \bar{\mu}^t(s_j)| \times \hat{\mathcal{N}}^t(s_j) \Big)}{|\mathcal{S}_i^t|}, \tag{5.6}$$

$$\mathbb{G}^t(s_j) = \frac{\sum\limits_{s_j \in \mathcal{U}_j^t} \Big(|\mathcal{Q}^t(i,j) - \mu^t(s_j) - \bar{\mu}^t(u_i)| \times \hat{\mathcal{N}}^t(u_i) \Big)}{|\mathcal{U}_j^t|} \tag{5.7}$$

where, $\mathcal{S}_i^t$ represents the set of services invoked by u_i at time-step t, and $\mathcal{U}_j^t$ represents the set of users that invoked service s_j at t. The mean-centered QoS values for u_i and

s_j, denoted by $\bar{\mu}^t(s_j)$ and $\bar{\mu}^t(u_i)$ respectively, are defined as follows:

$$\bar{\mu}^t(s_j) = \frac{\sum\limits_{u_i \in \mathcal{U}_j^t} \left(\mathcal{Q}^t(i,j) - \max\limits_{u_i \in \mathcal{U}_j^t} \mathcal{Q}^t(i,j) - \min\limits_{u_i \in \mathcal{U}_j^t} \mathcal{Q}^t(i,j) \right)}{|\mathcal{U}_j^t| - 2}, \tag{5.8}$$

$$\bar{\mu}^t(u_i) = \frac{\sum\limits_{s_j \in \mathcal{S}_i^t} \left(\mathcal{Q}^t(i,j) - \max\limits_{s_j \in \mathcal{S}_i^t} \mathcal{Q}^t(i,j) - \min\limits_{s_j \in \mathcal{S}_i^t} \mathcal{Q}^t(i,j) \right)}{|\mathcal{S}_i^t| - 2} \tag{5.9}$$

Additionally, $\hat{\mathcal{N}}^t(u_i)$ and $\hat{\mathcal{N}}^t(s_j)$ represent the normalized standard deviation for u_i and s_j at time-step t, as shown in Eq 5.10.

$$\hat{\mathcal{N}}^t(u_i) = 1 - \left(\frac{\sigma^t(u_i) - \min\limits_{u_k \in \mathcal{U}} \sigma^t(u_k)}{\max\limits_{u_k \in \mathcal{U}} \sigma^t(u_k) - \min\limits_{u_k \in \mathcal{U}} \sigma^t(u_k)} \right), \tag{5.10}$$

$$\hat{\mathcal{N}}^t(s_j) = 1 - \left(\frac{\sigma^t(s_j) - \min\limits_{s_k \in \mathcal{S}} \sigma^t(s_k)}{\max\limits_{s_k \in \mathcal{S}} \sigma^t(s_k) - \min\limits_{s_k \in \mathcal{S}} \sigma^t(s_k)} \right) \tag{5.11}$$

where, $\sigma^t(u_i) = sd(\mathcal{Q}^t(i,.))$ and $\sigma^t(s_j) = sd(\mathcal{Q}^t(.,j))$ represent the standard deviation of the QoS vectors corresponding to u_i and s_j at time-step t, respectively.

A high value of $\hat{\mathcal{N}}^t(u_i)$ or $\hat{\mathcal{N}}^t(s_j)$ generally indicates a consistent QoS invocation pattern for u_i or s_j at time-step t. Therefore, when calculating the GDI of a user u_i in relation to its individual service invocation s_j, we use $\hat{\mathcal{N}}^t(s_j)$ as a weight. This weight reflects the consistency of s_j and helps to determine whether an abnormal value is attributable to u_i or s_j. Similarly, the GDI for s_j is calculated. A high GDI value for u_i or s_j indicates a significant abnormality. Based on these GDI values, we labeled the grey-sheep users and services.

5.1.3.1.2 Grey-sheep Labeling. We define the grey-sheep indicator tensor $\mathcal{G}$, with dimensions $N \times 1 \times \mathcal{T}$, as follows:

$$\mathcal{G}(i,1,t) = \begin{cases} 1 & \text{if } i \leqslant n \text{ and } \mathbb{G}^t(u_i) > \mu_{gu}^t + c_1 * \sigma_{gu}^t \\ 1 & \text{if } i > n \text{ and } \mathbb{G}^t(s_i) > \mu_{gs}^t + c_2 * \sigma_{gs}^t \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

where, μ_{gu}^t and σ_{gu}^t represent the mean and standard deviation of the GDI values for users, while μ_{gs}^t and σ_{gs}^t represent the mean and standard deviation of the GDI values for services at time-step t. The positive constants c_1 and c_2 are two tunable hyperparameters. This tensor is then utilized in subsequent operations to obtain fine-tuned features for both users and services. In the next subsection, we introduce our next module, LPAM, which extracts user- and service-specific features that are particularly beneficial for grey-sheep instances, aiming to enhance prediction accuracy.

5.1.3.2 Local Pattern Adaptation Module (LPAM)

The objective of this module is to extract local patterns from the QoS invocation profiles of each user and service. For each time-step t within the $\mathcal{T}$-time-steps, the LPAM is used to obtain local features for each user $u_i \in \mathcal{U}$ and service $s_j \in \mathcal{S}$ based on their QoS profiles $\mathcal{Q}^t(i,.,t)$ and $\mathcal{Q}^t(.,j,t)$. These local features consist of 14 basic statistical metrics, including minimum, maximum, mean, median, standard deviation, skewness, kurtosis, interquartile range, mean absolute deviation, median absolute deviation, root mean square, absolute energy, entropy, and peak-to-peak difference [162]. These features provide diverse quantitative insights into the central tendencies, variability, and distributions of the data. We store these features in a tensor $\mathcal{X}_2$ with dimensions $N \times 14 \times \mathcal{T}$. In this tensor, the first n entries of the first dimension correspond to users, while the remaining m entries correspond to services. We then leverage $\mathcal{X}_2$ to manage GSU and GSU. In the next module, we will explain the process of injecting these features into the grey-sheep instances.

5.1.3.3 Localized Pattern Enhancement Module (LPEM)

Since collaborative features alone are insufficient to fully represent the GSU and GSU, we enhance them with profile-specific features to support representation learning and improve QoS prediction performance. To achieve this, we concatenate the collaborative features, $\mathcal{Y}_1$, with profile-specific features, $\mathcal{X}_2$, along the second dimension of the tensor, resulting in $\mathcal{Y}_2$ with dimensions $N \times (f_2+14) \times \mathcal{T}$. $\mathcal{Y}_2$ is then processed through a fully connected layer to reshape the tensor, generating the output tensor $\mathcal{Y}_3$ with

dimensions $N \times f_2 \times \mathcal{T}$.

To obtain the embedding for grey-sheep instances, we introduce a function f_g, as described in Eq. 5.13, which takes the grey-sheep indicator tensor $\mathcal{G}$ and $\mathcal{Y}_3$ to filter out features unique to grey-sheep instances from regular users and services. This results in the embedding $\mathcal{Z}_1 = f_g(\mathcal{G}, \mathcal{Y}_3)$.

For non-greysheep users and services, indicated by $\tilde{\mathcal{G}} = (\mathbb{1}_{N \times 1 \times \mathcal{T}} - \mathcal{G})$, only $\mathcal{Y}_1$ is used, leading to the embedding $\mathcal{Z}_2 = f_g(\tilde{\mathcal{G}}, \mathcal{Y}_1)$. Finally, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are combined to form $\mathcal{Z}_3$, which retains the distinguishing features of both grey-sheep and non-greysheep instances. This $\mathcal{Z}_3$ is subsequently used in the next module.

$$f_g(\mathcal{G}, \mathcal{Y}) = [\mathcal{Z}^1 \mid\mid \mathcal{Z}^2 \mid\mid \ \ldots \ \mid\mid \mathcal{Z}^{\mathcal{T}}]; \quad \forall t = [1..\mathcal{T}], \mathcal{Z}^t = (\mathcal{G}^t e) \odot \mathcal{Y}^t \tag{5.13}$$

where, e is a $1 \times f_2$ sized vector of 1s, $\odot$ denotes the Hadamard product, and $||$ represents channel-wise concatenation.

5.1.4 Temporal Granularity Extraction Module (TGEM)

The features obtained from the previous modules are limited in their ability to capture the intricate temporal dynamics necessary for accurate temporal QoS prediction. To address this, we introduce the module TGEM, designed to capture temporal features across different scales. TGEM consists of three primary components: (a) *E-block*: A transformer encoder block [130] that includes multi-head attention (MHA) mechanisms to capture long-range dependencies, (b) *T-block*: A block dedicated to capturing fine-grained temporal features, which is composed of two 1D convolutional layers for localized temporal feature extraction, and (c) *F-block*: A block for extracting fine-grained features at each individual time-step, consisting of two fully connected dense layers. These components, as illustrated in Fig. 5.2(c), work together to capture temporal features at varying levels of granularity.

5.1.4.1 Temporal Dependency Extractor Block (E-block)

This block is equipped with a transformer encoder [130], incorporating MHA mechanisms. The E-block applies multiple attention heads in parallel, followed by a fully

connected dense layer to capture multiple contexts from the input embedding. Additionally, it incorporates positional encoding to overcome the limitation of MHA in preserving the sequential information of the input data. We adopt sinusoidal positional encoding, similar to [130], to maintain temporal sequence information.

Upon receiving the input $\mathcal{Z}_3$ from GMM, batch normalization (BN) is applied to re-center and re-scale the features, reducing internal covariate shift and improving training stability and speed [163]. After calculating the temporal positional encoding (TPE) for each user-service feature at each time-step, we add the TPE to $BN(\mathcal{Z}_3)$ to produce the intermediate result $\mathcal{Z}_4$, which is then passed to the MHA block. However, the classical transformer encoder faces scalability challenges with high-dimensional feature embeddings. To address this, rather than passing the combined results directly to the MHA block, we introduce a fully connected dense layer, followed by another batch normalization step. This reduces the dimensionality of the input features by one-fourth, resulting in $\mathcal{Z}_5$ with dimension $N \times (f_2/4) \times \mathcal{T}$.

5.1.4.1.1 Multi-Head Attention (MHA). $\mathcal{Z}_5$ contains the features for all users and services across all $\mathcal{T}$ time-steps. For each user or service instance i (out of the N instances), the features across the $\mathcal{T}$ time-steps, denoted as $\mathcal{Z}_5^{[i]}$, are rearranged and concatenated row-wise to form the feature matrix $\mathcal{F}_i$, with dimensions $\mathcal{T} \times (f_2/4)$. This matrix is then used as input for the MHA that operates in three steps as follows: First, for each j^{th} head, we generate three different representations of $\mathcal{F}_i$, namely, the Query $(Q_a)_j = \mathcal{F}_i \mathcal{W}_j^{1[i]}$, the Key $(K_a)_j = \mathcal{F}_i \mathcal{W}_j^{2[i]}$, and the Value $(V_a)_j = \mathcal{F}_i \mathcal{W}_j^{3[i]}$, where $\mathcal{W}_j^{k[i]}$ (for $k \in 1, 2, 3$) are learnable weights. For simplicity, we refrain from writing bias terms in this representation. In the second step, we compute the scaled-dot product attention using $(Q_a^{[i]})_j$, $(K_a^{[i]})_j$, and $(V_a^{[i]})_j$ for each j^{th} head, as shown in Eq. 5.14, where d_k represents the dimension of the key used to scale dot-product. $(H_a^{[i]})_j$ accommodates self-attentive output embedding for each head j.

$$(H_a^{[i]})_j((Q_a^{[i]})_j, (K_a^{[i]})_j, (V_a^{[i]})_j) = softmax\left(((Q_a^{[i]})_j \cdot ((K_a^{[i]})_j)^T)/\sqrt{d_k}\right)(V_a^{[i]})_j \quad (5.14)$$

We employ total N_h numbers of heads in E-block. In the third and final step, the output embeddings from all heads are concatenated, followed by a linear transformation

to produce the output,

$$\mathcal{Z}_6^{[i]} = MHA((Q_a^{[i]})_j, (K_a^{[i]})_j, (V_a^{[i]})_j) = \left((H_a^{[i]})_1 \,||\, (H_a^{[i]})_2 \,||\, \ldots \,||\, (H_a^{[i]})_{N_h}\right) \mathcal{W} \tag{5.15}$$

where, $\mathcal{W}$ is learnable weight. Each head captures a specific temporal feature, and their combination captures the different contexts within the input embedding. Finally, $\mathcal{Z}_6^{[i]}$ is reshaped and combined across all user-service instances to form $\mathcal{Z}_6$. After the MHA operation, $\mathcal{Z}_6$ is passed through a dense layer to restore its original dimensions, $N \times f_2 \times \mathcal{T}$. This restored tensor is then added to $\mathcal{Z}_5$ using a residual connection, which preserves the collaborative characteristics for users and services, resulting in $\mathcal{Z}_7$ (refer to E-block of TGEM in Fig. 5.2(c)). $\mathcal{Z}_7$ is then passed through batch normalization, producing $\mathcal{Z}_8$ that is subsequently fed into the T-block and F-block. These blocks are explained next to obtain multi-granularity features from $\mathcal{Z}_8$.

5.1.4.2 Fine-Grained Temporal Analyzer Block (T-block)

To capture fine-grained temporal features, we employ the T-block, which utilizes two 1D convolutional layers (Conv1D) with a dropout layer [135] between them, applied along the temporal dimension of $\mathcal{Z}_8$. The first Conv1D layer, with $\mathcal{T}/4$ filters, performs temporal pooling, reducing the temporal dimension by a factor of four, resulting in an output of $N \times f_2 \times (\mathcal{T}/4)$. This step ensures that fine-grained features are effectively extracted. The second Conv1D layer, with $\mathcal{T}$ filters, restores the temporal dimension to match that of $\mathcal{Z}_8$. Finally, we apply a residual connection by adding $\mathcal{Z}_8$ to the output, forming the final output $\mathcal{Z}_T$.

5.1.4.3 Time-Step Feature Refiner Block (F-block)

Parallel to the T-block, the F-block is designed to capture coarse-grained features. It employs two fully connected dense layers applied over the feature dimension (i.e., the second dimension of the three-dimensional tensor) of the input, with a dropout layer placed between them. In the F-block, the input tensor $\mathcal{Z}_8$ is reshaped, interchanging the temporal and feature dimensions. The first dense layer applies a nonlinear trans-

formation that reduces the feature dimension to one-fourth, while the second dense layer restores it to its original dimension through another nonlinear transformation. The output from the second layer is reshaped back and added to $\mathcal{Z}_8$ via a residual connection from the E-block, resulting in the final output embedding, $\mathcal{Z}_F$.

Finally, we apply soft-attention over $\mathcal{Z}_T$ and $\mathcal{Z}_F$ to refine the temporal dynamics, yielding $\mathcal{X}_3$. It is important to note that while the parallel T-block and F-block enable multi-granularity feature extraction, the E-block captures various meaningful contexts. The output of the TGEM, $\mathcal{X}_3$, is then passed to the comprehensive QoS prediction module for final prediction.

5.1.5 Comprehensive QoS Prediction Module (CQPM)

The CQPM predicts the QoS of a user–service pair at time-step $\mathcal{T}$. It fuses the feature embeddings $\mathcal{Z}_3$ (from GMM) and $\mathcal{X}_3$ (from TGEM) via feature addition followed by batch normalization. Mean pooling aggregates historical features across the previous $\mathcal{T}$ steps, and the result is passed through two fully connected layers. The final layer output is partitioned row-wise into user and service feature matrices, $\mathscr{U}_{n\times f_4}$ and $\mathscr{S}_{m\times f_4}$. The QoS estimate is then obtained via their inner product:

$$\hat{\mathcal{Q}}^{\mathcal{T}} = \mathscr{U}_{n\times f_4} \cdot \mathscr{S}^{T}_{m\times f_4} \tag{5.16}$$

5.1.6 Training and Prediction

To address the issue of outliers, we adopt two strategies. First, we adopt the Cauchy loss, denoted by $\mathcal{L}$, as the training objective. The Cauchy loss, as shown in Eq. 5.17, is more robust to outliers compared to standard error metrics.

$$\mathcal{L} = \frac{1}{|\psi|} \sum_{\mathcal{Q}^{\mathcal{T}}(i,j)\in\psi} \log\left(1 + \left(\left(\mathcal{Q}^{\mathcal{T}}(i,j) - \hat{\mathcal{Q}}^{\mathcal{T}}(i,j)\right)/\gamma\right)^2\right) \tag{5.17}$$

where, ψ represents the set of valid entries in $\mathcal{Q}^{\mathcal{T}}$, and γ is a scale hyper-parameter. Second, we apply the unsupervised Isolation Forest algorithm [109] to detect and remove a fixed percentage of outliers, controlled by the hyperparameter λ, highlighting

HCTN's performance without outlier interference. AdamW optimizer [137] is used here to train our model.

5.1.7 Complexity Analysis

Let, n and m denote the number of users and services, respectively, and $N = n+m$ the total number of nodes. Let $\mathcal{T}$ represents the number of temporal snapshots, and $E = \sum_{t=1}^{\mathcal{T}} e_t$ denote the total number of observed QoS interactions across all $\mathcal{T}$ time steps. The asymptotic computational cost of each component of HCTN is summarized below:

- *Non-negative Matrix Decomposition (NMD):* Decomposing each $Q^t \in \mathbb{R}^{n\times m}$ into latent matrices $X_u^t \in \mathbb{R}^{n\times f_1}$ and $X_s^t \in \mathbb{R}^{m\times f_1}$ costs $\mathcal{O}(nmf_1)$ per iteration. Over I_{NMD} iterations and $\mathcal{T}$ time steps: $\mathcal{O}(I_{\text{NMD}} \cdot \mathcal{T} \cdot nmf_1)$.

- *Hypergraph Collaborative Filtering Module (HCFM):* Each snapshot employs an HCN with one dense projection $\mathcal{O}(Nf_1f_2)$ and l hypergraph-convolution layers, each costing $\mathcal{O}(Nf_2^2 + e_tf_2)$. Summed across all $\mathcal{T}$ snapshots: $\mathcal{O}(N\mathcal{T}f_1f_2 + N\mathcal{T}lf_2^2 + lf_2E)$.

- *Grey-sheep Mitigation Module (GMM):* Computing local statistical features scales as $\mathcal{O}(E)$, while dense fusion across $\mathcal{T}$ time steps costs $\mathcal{O}(\mathcal{T}Nf_2^2)$: $\mathcal{O}(E + N\mathcal{T}f_2^2)$.

- *Temporal Granularity Extraction Module (TGEM):* The multi-head attention (E-block) requires $\mathcal{O}(N\mathcal{T}^2f_2)$, and the subsequent dense transformations (T- and F-blocks) add $\mathcal{O}(N\mathcal{T}f_2^2)$: $\mathcal{O}(N\mathcal{T}^2f_2 + N\mathcal{T}f_2^2)$.

- *Collaborative QoS Prediction Module (CQPM):* Normalization and pooling incur $\mathcal{O}(N\mathcal{T}f_2)$, dense projections contribute $\mathcal{O}(Nf_2f_3 + Nf_3f_4)$, and final QoS reconstruction $US^\top$ adds $\mathcal{O}(nmf_4)$: $\mathcal{O}(N\mathcal{T}f_2 + Nf_2f_3 + Nf_3f_4 + nmf_4)$.

Combining the all components, the total asymptotic cost of HCTN is:

$$\mathcal{O}\left(I_{\text{NMD}} \cdot \mathcal{T} \cdot nmf_1 + l\mathcal{T}Nf_2^2 + lf_2E + N\mathcal{T}^2f_2 + N\mathcal{T}f_2^2 + nmf_4\right).$$

In sparse QoS settings ($E \ll N^2$) and small $\mathcal{T}$, the runtime is dominated by the hypergraph convolution term $\mathcal{O}(lf_2E)$ and the dense transformation term $\mathcal{O}(N\mathcal{T}f_2^2)$, ensuring linear scalability with the number of observed interactions and maintaining efficient training and inference.

5.2 Experiments

HCTN was trained offline using TensorFlow 2.16.1 with Python 3.10.13 on an NVIDIA GeForce RTX 4080 GPU. For performance evaluation, the model was tested on a system equipped with an AMD Ryzen-9 7950X 16-core processor and 32 GB RAM.

5.2.1 Experimental Setup

Train-Validation-Test Split. Our method is evaluated on WSDREAM-2 [127] dataset for two QoS parameters, RT and TP, illustrated in Sec. 2.3. The results were reported under five training densities $\psi \in \{5, 10, 15, 20, 50\}$ which are indexed as RT-ψ and TP-ψ Since the datasets are temporal, we retain the first τ time-steps exactly as they appear in the original QoS tensor to serve as the historical input sequence. For the final $(\tau + 1)^{\text{th}}$ time-step, we randomly sample $\psi\%$ of the valid QoS entries for training and use the remaining $(100 - \psi)\%$ entries for testing; additionally, 20% of the sampled training entries are reserved for validation. This sampling procedure is repeated five times for each ψ, and the reported results correspond to the average across these five independent train–test–validation splits, ensuring robustness against randomness and varying sparsity levels.

Hyperparameter Configuration. Table 5.1 details the hyperparameters used for

Table 5.1: Hyperparameter settings of HCTN

Hyperparameter	Description	Values
$\mathcal{T}$	Time Window	{2, 4, 6, 8, 10, 12, 14}
l	HCN layers	{1, 2, 3, 4}
N_h	No. of heads	{1, 2, 3, 4, 5}
d_k	Head size	{32, 64, 128, 256, 512}

experimental analysis in the main manuscript. We aim to optimize model performance across various configurations by systematically adjusting these hyperparameters through an exhaustive grid search over the specified parameter ranges, conducted on the validation set using a five-fold cross-validation strategy. The configuration achieving the lowest average error in terms of MAE and RMSE was selected as the final model setting. During hyperparameter tuning, each configuration was trained for up to $20K$ epochs using an AdamW optimizer [137] with a stepwise learning rate schedule. The learning rate was initialized at 10^{-3} and progressively reduced to facilitate smoother convergence and prevent premature stagnation. Early stopping (patience = 250) was employed to avoid overfitting and ensure stable training.

5.2.2 Performance Analysis

We now present an analysis of the performance of HCTN, starting with a comparison between HCTN and previous temporal QoS prediction methods.

5.2.2.1 Comparison with Existing Approaches

5.2.2.1.1 Performance Comparison of HCTN with Past Methods Table 5.2 displays the performance of HCTN on the RT and TP datasets across five different training densities. HCTN significantly outperformed major previous methods in terms of both MAE and RMSE. The overall performance improvement relative to the second-best method, denoted as PG (%), is detailed in Table 5.2. Additionally, the results show HCTN performance improved with increased training density.

5.2.2.1.2 Performance Comparison After Outlier Removal We compared HCTN with all sixteen SOTA methods after removing 10% of outliers (i.e., λ=10) from the RT and TP datasets at three different training densities, as shown in Table 5.3. HCTN consistently achieved the lowest MAE and RMSE, demonstrating its superior predictive accuracy and robustness. This performance improvement is especially evident under challenging conditions, such as low training density, where the

Table 5.2: Performance comparison of HCTN with previous methods

Methods	Loss Function	MAE					RMSE				
		RT-5	RT-10	RT-15	RT-20	RT-50	RT-5	RT-10	RT-15	RT-20	RT-50
WSPred [17]	MSE	2.5580	2.4990	2.4100	2.3000	2.1266	4.3626	4.2892	4.2000	4.1500	3.8943
TUIPCC [62]	-	2.4088	2.2118	2.0910	1.9531	1.2636	4.8319	4.7751	4.6329	4.4454	3.1722
GFEN [19]	MSE	2.3168	2.2901	2.2614	2.2968	1.9395	5.5295	5.4669	5.3925	5.1024	4.9139
SCATSF [14]	Huber	2.2657	2.2203	2.1234	2.0273	1.6352	5.4799	5.2403	4.9461	4.9326	3.9418
NNCP [116]	MSE	1.8000	1.7500	1.5500	1.5800	1.3000	3.9000	4.0500	3.7500	3.5000	2.9000
TRCF [61]	-	1.7574	1.3319	1.1229	1.0132	0.8714	4.7187	3.9072	3.4028	3.0563	2.3592
DeepTSQP [18]	MSE	1.7329	1.5241	1.4540	1.3137	0.9710	4.1902	3.8777	3.5964	3.1313	2.3576
PLMF [118]	MSE	1.4338	1.3031	1.2439	1.2209	1.1533	2.9201	2.7817	2.7185	2.6946	2.6278
STGCN [164]	MSE	1.3995	1.3088	1.2642	1.2308	1.1956	3.5544	3.4332	3.3461	3.3011	3.2512
RNCF [139]	MSE	1.3386	1.2713	1.2069	1.1583	1.0762	3.1369	2.8542	2.7789	2.6038	2.6038
TaTruSR [20]	-	1.3029	1.1195	1.0644	1.0253	0.9462	3.4266	2.9562	2.7738	2.6927	2.5351
CTF [42]	Cauchy	1.2884	1.1753	1.1504	1.0947	1.0430	2.9253	2.7172	2.6368	2.5769	2.4797
BNLFT [99]	MSE	1.2600	1.2400	1.2300	1.2200	1.2000	2.8000	2.5000	2.4800	2.4500	2.4000
GMCL [70]	MAE + CL	1.1572	1.1058	1.0748	1.0294	0.9542	3.3473	3.3285	3.1860	3.0952	2.9768
STGFT [165]	MAE	1.1098	0.8812	0.7265	0.6512	0.4536	3.1857	2.9613	2.6520	2.4008	2.2116
TPMCF [159]	Cauchy	0.9735	0.8132	0.7168	0.6159	0.2528	2.9648	2.6856	2.6191	2.2927	1.5832
HCTN	**Cauchy**	**0.8453**	**0.6765**	**0.6133**	**0.5256**	**0.2237**	**2.4486**	**2.2336**	**2.1074**	**1.9754**	**1.2853**
PG (%)		13.17%	16.81%	14.44%	14.66%	11.51%	12.55%	10.66%	15.02%	13.84%	18.82%
		TP-5	TP-10	TP-15	TP-20	TP-50	TP-5	TP-10	TP-15	TP-20	TP-50
TUIPCC [62]	-	15.2948	14.9539	14.0606	13.5154	8.8786	44.1169	43.8122	43.7239	43.1728	33.2959
GFEN [19]	MSE	13.8755	13.8538	13.7270	14.3370	12.9108	60.3329	58.4838	60.2614	60.1972	59.4645
SCATSF [14]	Huber	13.1707	12.3124	11.9334	11.0718	9.2442	59.8496	58.5441	57.7450	56.4208	52.4242
TaTruSR [20]	-	11.1396	9.0643	8.2291	7.6819	6.6458	52.1592	42.1581	39.7379	38.0506	33.0270
DeepTSQP [18]	MSE	9.6616	8.3258	8.8682	9.3145	10.4283	49.4513	43.1617	41.1476	38.9401	36.3836
PLMF [118]	MSE	9.2414	9.2217	9.1666	9.1091	9.1751	51.7130	51.6902	51.3862	51.1958	51.8412
TRCF [61]	-	9.1978	6.4491	4.8417	4.0694	3.1343	47.8063	38.8540	31.0267	28.0598	20.5362
WSPred [17]	MSE	8.2761	8.0131	7.8500	7.6000	6.8000	39.0962	38.6251	38.5000	37.6000	36.5724
TPMCF [159]	Cauchy	7.0860	5.7447	4.8590	4.5136	1.4232	40.1241	37.6999	34.7010	30.5240	20.7425
RNCF [139]	MSE	6.4916	5.6454	5.2319	5.4083	4.5919	32.7867	28.4236	26.8409	25.3949	22.8516
STGCN [164]	MSE	5.5987	4.9839	5.0139	4.5411	3.9105	46.9290	44.9156	42.7702	42.5804	38.0443
GMCL [70]	MAE + CL	5.3213	4.4768	4.4021	4.1963	4.1603	36.9408	31.1152	30.2974	27.7784	26.7878
NNCP [116]	MSE	5.1000	4.5000	4.2500	4.1000	3.7500	32.5000	32.5000	25.5000	24.5000	23.5000
STGFT [165]	MAE	4.3302	4.2174	3.9836	3.7692	3.2256	28.5001	26.4642	24.7985	23.2439	22.1006
BNLFT [99]	MSE	4.2500	4.1000	3.9000	3.7500	3.4000	24.5000	23.4000	21.5000	22.6000	19.9000
CTF [42]	Cauchy	4.0279	3.6458	3.5602	3.5578	3.4856	32.0197	31.6197	31.2808	31.4697	28.6353
HCTN	**Cauchy**	**3.9693**	**3.5486**	**3.3071**	**2.8671**	**0.7404**	**23.8219**	**21.3090**	**20.8300**	**18.8000**	**4.8526**
PG (%)		1.45%	2.67%	7.11%	19.41%	47.98%	2.77%	8.94%	3.12%	16.81%	75.62%

Table 5.3: Comparative study of HCTN with $\lambda = 10$

Methods	RT						TP					
	MAE			RMSE			MAE			RMSE		
	10	20	50	10	20	50	10	20	50	10	20	50
WSPred [17]	2.1743	1.8934	1.8279	3.8617	3.6859	3.5229	6.8200	6.2566	5.8448	34.7757	33.3952	33.0842
TUIPCC [62]	1.8622	1.6273	1.0306	4.4744	4.1678	2.9312	6.3668	5.5349	4.0549	13.6695	11.3400	9.0899
GFEN [19]	1.7787	1.7775	1.5139	5.0724	4.9139	4.6031	3.6144	3.4130	2.3500	5.6124	5.4232	4.4911
SCATSF [14]	1.6076	1.5661	1.3333	4.8608	4.5484	3.6764	2.4281	2.1165	1.8738	4.9687	4.0603	3.6505
DeepTSQP [18]	1.1732	1.0382	0.7216	3.5343	2.8141	1.9867	2.4883	4.1765	5.3630	5.7691	7.7521	9.4549
PLMF [118]	1.1060	1.0316	0.9679	2.5411	2.4606	2.4003	2.4712	2.2602	2.3924	3.6705	3.8363	3.8347
BNLFT [99]	1.0828	1.0575	1.0475	2.6181	2.5809	2.5659	1.4241	1.3935	1.3319	4.6031	4.4685	4.4319
NNCP [116]	1.0796	1.0536	1.0521	2.6401	2.5797	2.5805	1.5079	1.4342	1.3926	4.9207	4.7019	4.5026
RNCF [139]	1.0530	0.9482	0.8874	1.9867	2.4876	2.3733	1.8907	1.4847	1.5720	5.2260	4.8451	4.8397
TRCF [61]	1.0354	0.7975	0.6995	3.6268	2.8328	2.1605	1.4895	1.0227	0.8414	4.9592	4.6396	3.9317
CTF [42]	0.9215	0.8981	0.8879	2.5865	2.5579	2.5541	1.3567	1.1945	1.0193	3.0436	2.9225	2.7989
STGCN [164]	0.8663	0.8209	0.8191	2.8327	2.7455	2.7256	1.8283	1.4947	1.1833	4.4639	3.5102	2.8747
GMCL [70]	0.8517	0.7860	0.7183	2.5405	2.4924	2.4723	1.1780	1.1545	1.1222	3.7307	3.7236	3.5349
TaTruSR [20]	0.7742	0.6991	0.6316	2.4738	2.2084	2.0554	3.0775	2.6818	2.2956	13.1061	10.3942	8.2938
STGFT [165]	0.5728	0.4298	0.3266	2.3098	1.9686	1.8577	1.4126	1.2423	1.1034	3.1192	2.8443	2.6180
TPMCF [159]	0.4973	0.3864	0.1641	2.2709	1.8885	1.3623	1.0101	0.7881	0.2210	2.9564	2.9092	1.4372
HCTN	**0.4713**	**0.3549**	**0.1569**	**1.9017**	**1.6893**	**1.1082**	**0.9286**	**0.7204**	**0.2187**	**2.9294**	**2.8475**	**1.1385**
PG (%)	5.23	8.15	4.39	16.26	10.55	18.65	8.07	8.59	1.04	0.91	2.12	20.78

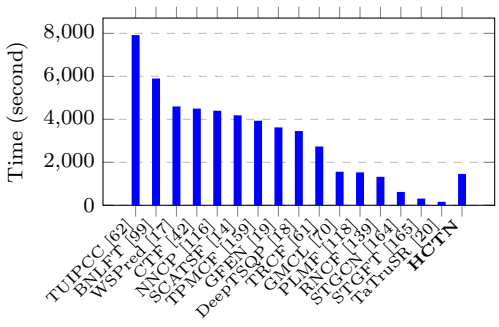


Figure 5.3: Training time comparison

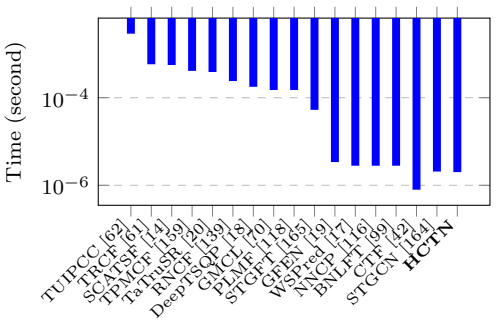


Figure 5.4: Inference time comparison

presence of outliers can severely distort the learning process, and removing even a small proportion of them leads to significant gains in prediction accuracy.

5.2.2.1.3 Training Time of HCTN HCTN was compared with all reported SOTA methods on training time, as shown in Fig. 5.3. Compared to TPMCF [159], CTF [42], and BNLFT [99], which previously achieved the best prediction accuracy across various cases, HCTN is faster by 1.87×, 3.04×, and 4.01×, respectively.

This demonstrates that HCTN can be retrained efficiently with minimal overhead as training density increases, resulting in enhanced prediction performance.

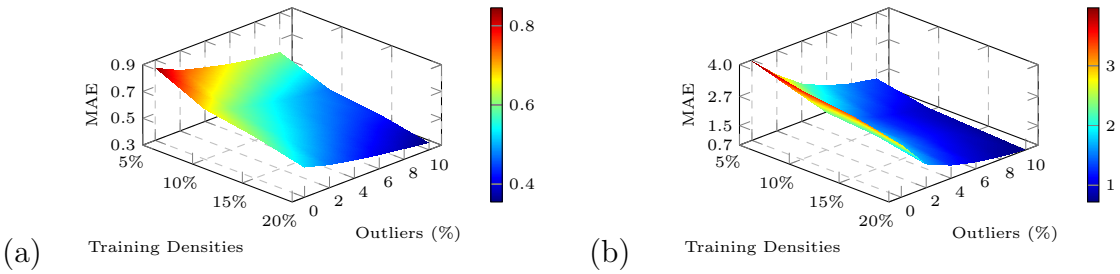


Figure 5.5: Impact of outliers on (a) D1, and (b) D2 datasets.

5.2.2.1.4 Inference Time of HCTN Real-time systems, especially those involving safety-critical applications, require swift QoS predictions. Fig. 5.4 shows that HCTN outperformed all the reported SOTA methods in terms of inference time. The average prediction time of HCTN was approximately 2×10^{-6} seconds, a negligible duration compared to the minimum response time of services (0.001 second), as indicated in Table 2.3. This demonstrates that our method is well-suited for integration into real-time applications.

Furthermore, while RNCF [139] exhibits a better training time compared to HCTN by 1.10×, it falls short in inference time, which is in the range of 2.4×10^{-4} (Fig.s 5.3 and 5.4). TaTruSR [20], being a memory-based method that employs only user similarity, also shows lower training time than HCTN. However, its inference time remains relatively high, in the order of 3.8×10^{-4}. In contrast, HCTN significantly outperformed both methods in terms of inference efficiency.

5.2.2.2 Performance in Under Anomalies Consideration

5.2.2.2.1 Impact of Outliers Fig.s 5.5(a) and (b) depict the performance of HCTN in terms of MAE across varying levels of outlier removal ($\lambda = 0$ to $\lambda = 10$, in 2% intervals) on the test datasets corresponding to RT and TP, with varying training densities. Key observations indicate that as λ increased, more outliers were detected and removed from the test datasets, leading to improved performance of HCTN. However, the rate of improvement gradually decreased across all cases. The initial 2% outlier removal yielded substantial performance gains, with average MAE improvements of 10.53% for the RT dataset and 34.80% for the TP dataset. In contrast, the

Table 5.4: Performance (MAE) of HCTN with selectively using LPAM with c_1=c_2=1

Datasets	$\lambda = 0$			$\lambda = 10$		
	HCTN − GMM	HCTN − GDM	**HCTN**	HCTN − GMM	HCTN − GDM	**HCTN**
RT-10	0.7036	0.6889	**0.6765**	0.5055	0.4861	**0.4713**
RT-20	0.5364	0.5330	**0.5256**	0.3669	0.3617	**0.3549**
TP-10	3.6713	3.5765	**3.5486**	0.9550	0.9522	**0.9286**
TP-20	2.9956	2.9190	**2.8671**	0.7461	0.7451	**0.7204**

HCTN − GMM: HCTN without grey-sheep Mitigation Module;
HCTN − GDM: HCTN without grey-sheep Detection Module, fully utilizing LPAM

performance gains became less pronounced in the final 2% of outlier removal (from 8% to 10%), with only 4.65% and 14.45% average improvements in MAE for RT and TP, respectively.

Analysis of the impact of outliers on HCTN performance showed that removing outliers significantly improved predictions, with the largest gains observed up to 6% removal. Beyond this point, improvements taper off, indicating diminishing returns as the data becomes cleaner.

5.2.2.2.2 Impact of grey-sheep HCTN leverages LPAM to extract local features for individual users and services, combining these with the global feature representations from HCFM to effectively manage grey-sheep instances. Table 5.4 shows that HCTN benefits from including local features for grey-sheep instances as opposed to excluding them (refer to HCTN - GMM in Table 5.4). On average, HCTN achieved a 2.93% and 3.82% improvement over HCTN-GMM on the RT and TP datasets, respectively. Furthermore, applying local features to all users and services (refer to HCTN - GDM in Table 5.4) degraded performance. HCTN showed an average improvement of 1.59% and 1.28% over HCTN-GDM on RT and TP, respectively. Although these improvements are modest, this is likely due to the small proportion of grey-sheep users (9.15%) and services (11.51%) identified, as only a limited fraction benefits from the GMM module. Nevertheless, addressing grey-sheep instances is critical for achieving more robust predictions in challenging cases. This analysis highlights the importance of the GMM in selectively integrating local features to enhance accuracy.

Fig.s 5.6(a) and (b) illustrate the impact of the number of grey-sheep instances

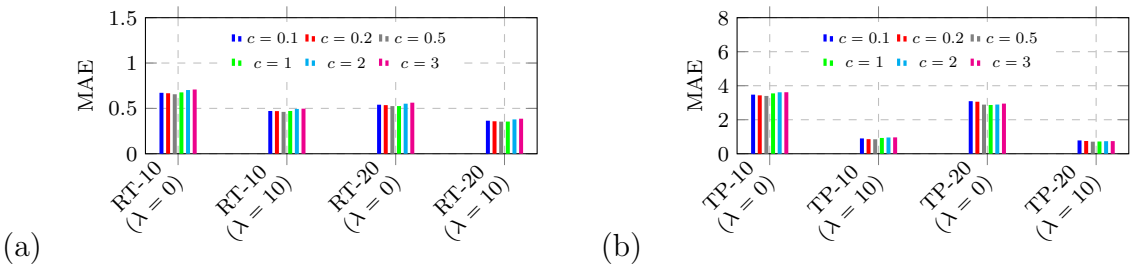


Figure 5.6: Impact of grey-sheep on (a) RT, and (b) TP datasets.

detected by the GDM and handled by the LPEM, which enhances their feature representation from HCFM by incorporating local patterns of users and services. As the grey-sheep detection thresholds c_1 and c_2 decreased (here, $c_1 = c_2 = c$), the number of identified grey-sheep instances increased, initially resulting in improved HCTN performance. For example, when c decreased from 3 to 1, HCTN achieved an average MAE improvement of 5.56% on RT and 2.29% on TP, indicating that selectively enhancing more atypical users/services boosts prediction accuracy. However, further lowering the threshold beyond $c = 1$ (e.g., $c = 0.2, 0.1$) led to performance degradation. This was due to an excessive number of users and services being labeled as grey-sheep, which overwhelmed the model with local patterns and diluted the benefits of collaborative filtering. Thus, while grey-sheep-aware enhancement improves accuracy up to a point, over-identification of grey-sheep instances can harm performance. These findings highlight the importance of appropriately setting the grey-sheep detection parameters to balance global and local pattern modeling for robust prediction.

5.2.2.2.3 Impact of Cold-start To assess the impact of cold-start conditions on HCTN, we simulate controlled cold-start scenarios by randomly selecting a fixed percentage $\xi \in \{5, 10, 15, 20\}$ of users or services and removing all their historical QoS records. Three evaluation settings are constructed: (a) *Cold-start Users (CSU)*, where QoS histories for ξ% of users are removed; (b) *Cold-start Services (CSS)*, where QoS histories for ξ% of services are removed; and (c) *Cold-start Both (CSB)*, where QoS histories for ξ% of both users and services are removed. After applying the respective cold-start mask, the model is trained using the remaining valid entries

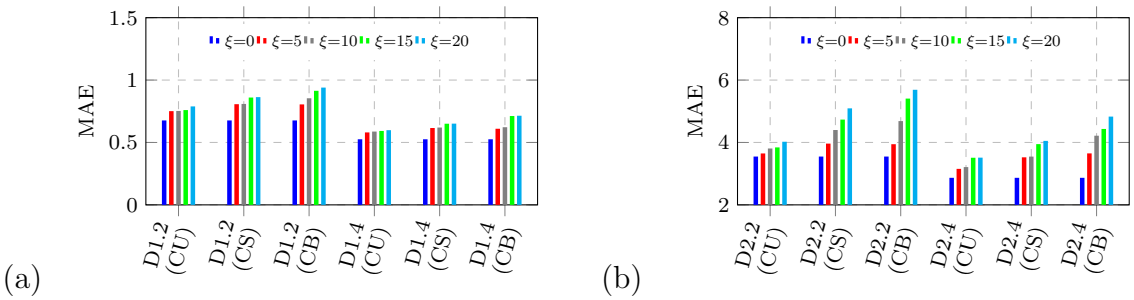


Figure 5.7: Performance of HCTN on cold-start with varying ξ.

following the same train–validation–test protocol. For each ξ and each scenario, the complete training and evaluation pipeline is repeated five times with different random selections, and the mean performance across the five runs is reported. Fig. 5.7 shows the results of these scenarios, and our key observations are summarized below:

(i) As ξ increased from 0 to 20, HCTN performance declined. Further, for the same value of ξ, the performance degradation was highest in the CSB scenario, followed by CSS; with CSU showing the least decline possibly due to the impact of service on the QoS value of a user-service pair is more prominent than that of a user.

(ii) As the training density increased, performance increased for all scenarios, CSU, CSS, and CSB, for the same value of ξ. This suggests that even with cold-starts in QoS logs, the effects can be reduced by increasing training density, supplementing the matrix factorization to generate better latent features to handle cold-start. This justifies the requirements of matrix factorization features.

To show the effectiveness of HCTN in addressing the cold-start problem, we compare it with two prominent past methods, CTF [42] and TPMCF [159], both handle temporal QoS prediction. CTF leverages tensor factorization to handle cold-start situations, while TPMCF uses non-negative matrix factorization to extract user/ service features, helping to alleviate the cold-start issue.

Table 5.5 presents a comparison of the CSB scenario across various cold-start percentages, $\xi = \{0, 5, 10, 15, 20\}\%$, on all datasets. On average, the performance gains of the HCTN over TPMCF on RT and TP datasets are 8.01% and 30.01%,

Table 5.5: Cold-start users and service comparison of HCTN with CTF and TPMCF

ξ	RT-10			RT-20			TP-10			TP-20		
	CTF	TPMCF	HCTN	CTF	TPMCF	HCTN	CTF	TPMCF	HCTN	CTF	TPMCF	HCTN
0	1.8908	0.8132	0.6765	1.5337	0.6159	0.5256	10.9274	5.7447	3.5486	8.6924	4.5136	2.8671
5	1.9283	0.8490	0.8050	1.6072	0.6945	0.6099	11.7714	7.0479	3.9443	9.4462	5.4211	3.6499
10	1.9462	0.8993	0.8541	1.6173	0.7069	0.6221	12.7103	7.2463	4.6851	9.9722	5.5479	4.2163
15	1.9538	0.9419	0.9140	1.6235	0.7259	0.7119	12.6588	7.2771	5.4062	10.0847	5.6973	4.4329
20	1.9624	0.9590	0.9397	1.6303	0.7708	0.7142	12.8640	7.3544	5.6886	10.4121	5.9460	4.8293

respectively, which shows the effectiveness of HCTN over TPMCF. The performance gain of HCTN with respect to CTF is even higher, highlighting the benefits of utilizing higher-order features. This performance boost in HCTN is due to its effective feature utilization in cold-start situations and its ability to model non-linear features.

5.2.2.3 Ablation Study

5.2.2.3.1 Impact of Different Modules in HCTN This analysis justifies the necessity of each module within HCTN by highlighting their contributions to overall performance, measured by MAE, across various λ values on the RT and TP datasets. As shown in Table 5.6, including HCFM (i.e., HCTN − TGEM) rather than excluding it (i.e., HCTN − HCFM − TGEM) resulted in an average improvement of 2.51% on RT and 6.11% on TP. Similarly, adding TGEM (i.e., HCTN − HCFM) rather than ablating it (i.e., HCTN − HCFM − TGEM) led to a 1.23% and 2.22% improvement on RT and TP, respectively. Combining both TGEM and HCFM (i.e., HTCN) yielded even better results, with an average improvement of 1.59% on RT and 5.44% on TP over the HCTN − TGEM setup. Furthermore, HCTN showed a notable improvement of 3.73% on RT and 8.43% on TP over HCTN − HCFM, highlighting the importance of the HCFM and TGEM modules in optimizing performance.

5.2.2.3.2 Feature Ablation Study: Impact of Higher-order Graphs Features This analysis highlights the significance of integrating higher-order collaborative features extracted from hypergraphs. HCFM is designed to extract features from FIG, as well as from SUIG and SSIG. FIG captures features from heterogeneous nodes, while SUIG and SSIG focus on homogeneous user-user and service-service node inter-

Table 5.6: Module ablation study for HCTN (MAE)

Datasets	Modules	λ					
		0	2	4	6	8	10
RT-10	HCTN – HCFM – TGEM	0.7075	0.6335	0.5896	0.5511	0.5223	0.5027
	HCTN – TGEM	0.6995	0.623	0.5803	0.5424	0.5129	0.4925
	HCTN – HCFM	0.7058	0.6299	0.5882	0.5507	0.521	0.5008
	HCTN	0.6765	0.6093	0.5670	0.5298	0.5008	0.4713
RT-20	HCTN – HCFM – TGEM	0.5559	0.4924	0.4577	0.4267	0.4029	0.3866
	HCTN – TGEM	0.5343	0.4892	0.4536	0.4221	0.3981	0.3814
	HCTN – HCFM	0.5436	0.4798	0.4437	0.4114	0.3869	0.3698
	HCTN	0.5256	0.473	0.4373	0.4054	0.3811	0.3549
TP-10	HCTN – HCFM – TGEM	4.1140	2.3369	1.6610	1.3194	1.1099	0.9642
	HCTN – TGEM	3.8710	2.3314	1.6557	1.3169	1.1018	0.9550
	HCTN – HCFM	3.9846	2.3325	1.6372	1.3002	1.0904	0.9474
	HCTN	3.5486	2.2463	1.6144	1.2820	1.0720	0.9286
TP-20	HCTN – HCFM – TGEM	3.0876	1.7567	1.3180	1.0327	0.8655	0.7562
	HCTN – TGEM	2.8926	1.7051	1.2810	1.0289	0.8565	0.7476
	HCTN – HCFM	3.0475	1.7269	1.2865	1.0461	0.8514	0.7446
	HCTN	2.8671	1.6767	1.2422	0.9876	0.8288	0.7204

actions, respectively. Fig. 5.8 illustrates that combining features from FIG, SUIG, and SSIG yielded superior performance compared to using FIG alone. The combined features resulted in an average improvement of 6.99% on the RT dataset and 19.06% on the TP dataset over the features extracted from FIG alone. Additionally, the combined features outperformed those extracted from SUIG and SSIG by an average of 1.28% on RT and 17.88% on TP, highlighting the enhanced value of complex feature integration enabled by the introduction of QIHG.

5.2.2.3.3 Feature Ablation Study: Impact of Multi-granularity Temporal Features This analysis demonstrates the value of integrating multi-granularity

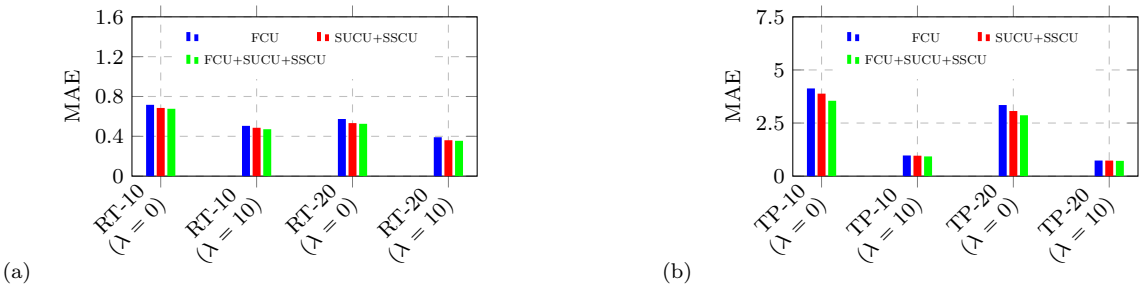


Figure 5.8: Impact of hypergraph on (a) RT, and (b) TP datasets.

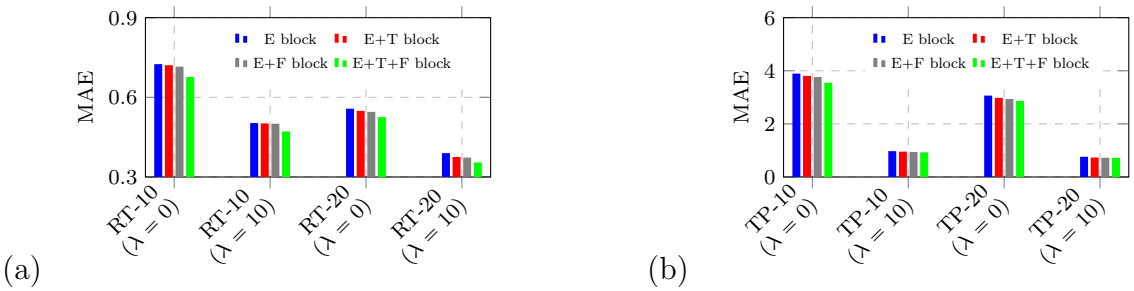


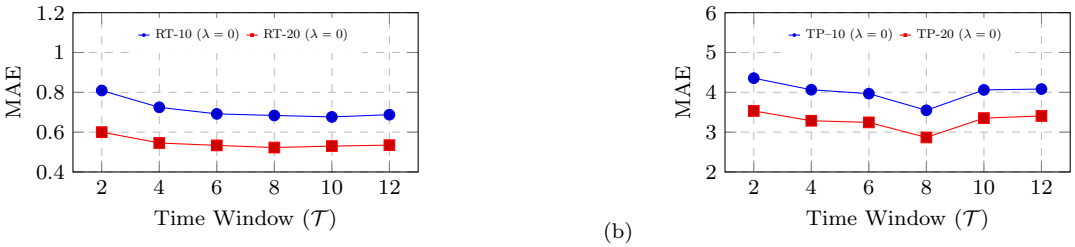
Figure 5.9: Impact of various blocks in TGEM on (a) RT, and (b) TP datasets.

temporal features using TGEM, which includes the E, T, and Fblock. The T block specializes in capturing fine-grained temporal features, while the F block focuses on coarse-grained features. Fig. 5.9 illustrates that combining both fine-grained and coarse-grained features yielded better performance on the RT and TP datasets compared to using each feature type individually. The combined features from the E + T + F blocks provided an average improvement of 6.58% on the RT dataset and 8.17% on the TP dataset over the features extracted by the E block alone. Similarly, E + T + F blocks resulted in a 5.52% and 5.63% improvement on the RT and TP datasets, respectively, compared to the E + T block setup. Furthermore, E + T + F blocks outperformed the E + F block combination by 4.53% on RT and 4.02% on TP, underscoring the more significant impact of incorporating more comprehensive and diverse feature sets.

5.2.2.4 Impact of Hyper-parameters

This section highlights the impact of different hyperparameters used in HCTN.

5.2.2.4.1 Impact of Time Window We explore the optimal number of time sequences needed for high QoS prediction performance. Fig.s 5.10(a)-(b) show the experiments by varying the time window (τ) from 2 to 12 with a step size of 2 for all datasets. The performance improves as τ increases for both datasets. At relatively lower τ, performance deteriorates, which is due to not incorporating adequate temporal features essentially required for the high prediction performance. Further, incorporating more time slices enhances QoS prediction up to a certain τ. Specifically,



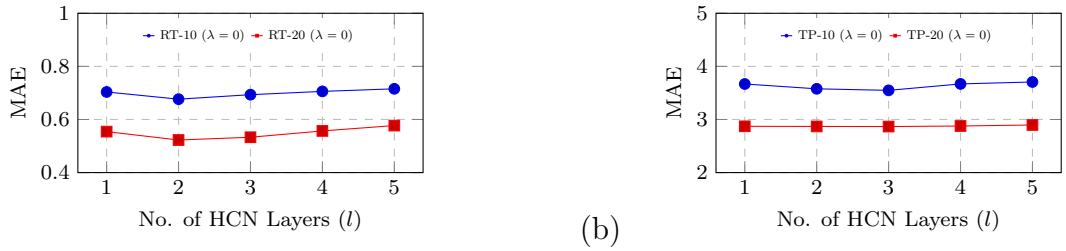
(a) (b)

Figure 5.10: Impact of number of time-windows ($\mathcal{T}$) on (a) RT, and (b) TP datasets.

for the RT-10 and RT-20 datasets, we achieved the highest prediction performance for $\tau = 10$ and $\tau = 8$, respectively. Similarly, for TP datasets, a common $\tau = 10$ achieve the competitive performance.

This experiment highlights that while a sufficient number of time slices is crucial for improving model performance by offering more time information, excessively large τ can negatively impact the model.

5.2.2.4.2 Impact of HCN Layers This experiment shows the utilization of higher-order features obtained using hypergraph collaborative filtering. Fig.s 5.11(a)-(b) illustrate the impact of the number of HCN layers (l) by varying from 1 to 5. We achieved competitive performance for $l = 2$ and $l = 3$ for RT and TP, respectively. However, for higher l, performance deteriorates since, with increased hypergraph convolution layer, the node features propagation may aggregated to give indistinguishable node embedding.



(a) (b)

Figure 5.11: Impact of no. of HCN layers (l) on (a) D1, and (b) D2 datasets.

5.2.2.4.3 Impact of Number of Heads and Head Size in MHA To optimize the parameters in multi-head attention (MHA), we investigate the sufficient require-

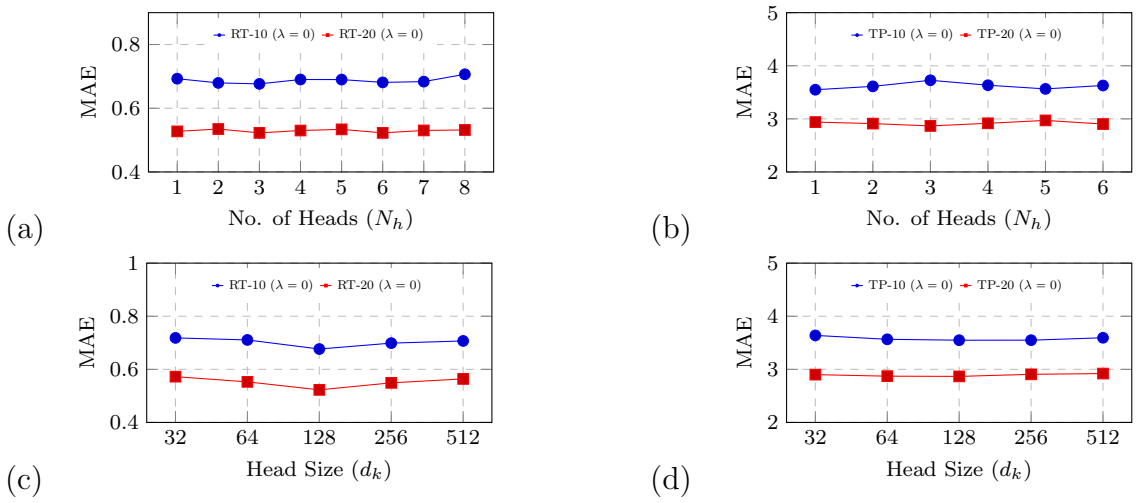


Figure 5.12: (a)-(b) Impact of no. of heads, and (c)-(d) head size

ments of the number of heads (N_h) and head size (d_k) affecting model performance. Our key observations are as follows.

(i) Utilizing multiple heads in MHA enables the capture of multi-dimensional information from the features. This experiment intends to find the optimal number of heads that balance prediction performance with the number of learnable parameters, since an increment in the number of heads also increases the learnable parameters. Fig.s 5.12(a)-(b) display the impact of N_h for all datasets. We achieved the best performance for $N_h = 3$ and $N_h = 6$ for RT-10 and RT-20, respectively. However, $N_h = 1$ for TP-10, and $N_h = 3$ for TP-20 worked better.

(ii) The head size (d_k) refers to the dimensionality of each attention head in MHA. Specifically, it controls the size of features each attention head incorporates. For a fixed input features dimension, a smaller d_k captures finer details, while larger heads capture broader information. To assess the head size, we vary d_k in powers of 2, ranging from 2^5 to 2^9. Fig.s 5.12(c)-(d) show the performance with different d_k for all datasets. We achieved optimal performance for d_k=2^7=128 across all datasets.

5.2.2.5 Memory and Computational Efficiency

This section presents a detailed complexity analysis of HCTN regarding model size and computational efficiency. We first quantify the implementation-level complexity by reporting the total number of trainable parameters and estimating the floating-point operations (FLOPs) required per forward pass. We then complement these practical measurements with an asymptotic analysis of each major module to characterize HCTN's theoretical computational behavior.

Together, these results provide a comprehensive assessment of HCTN's resource requirements and demonstrate its suitability for deployment under diverse hardware and memory constraints.

Table 5.7 summarizes the computational complexity of HCTN in comparison with two strong QoS prediction baselines, CTF [42] and TPMCF [159], by reporting both the number of trainable parameters and the floating-point operations (FLOPs) required per forward pass. The results indicate that HCTN achieved an effective balance between representational capacity and computational efficiency.

Specifically, HCTN contains 551.56K parameters, an 8.6× reduction relative to TPMCF, while requiring only 27.95 GFLOPs, which is 4.5× lower than TPMCF's computational cost. While CTF attained the smallest model size and the lowest FLOPs, its prediction accuracy on the WSDREAM QoS benchmark remained noticeably lower than that of HCTN. This suggests that CTF's highly compact design, although computationally attractive, may limit its ability to capture the richer spatio–temporal and collaborative patterns required for accurate QoS estimation. In contrast, HCTN offers a more favorable accuracy–efficiency trade-off: it delivers substantially improved prediction performance with only a modest increase in computation. Overall, the results

Table 5.7: Model parameter counts and FLOPs

Method	Params (K)	FLOPs (G)
CTF [42]	0.7059	1.8403
TPMCF [159]	4765.3880	126.6281
HCTN	551.5640	27.9539

Table 5.8: Statistical analysis using confidence intervals

CL	RT-10	RT-20	TP-10	TP-20
90%	(0.6812, 0.6904)	(0.5214, 0.5296)	(3.5424, 3.5548)	(2.8166, 2.9176)
95%	(0.6804, 0.6913)	(0.5206, 0.5304)	(3.5412, 3.5560)	(2.8069, 2.9273)
99%	(0.6786, 0.6930)	(0.5191, 0.5320)	(3.5389, 3.5584)	(2.7880, 2.9462)
Mean	0.6765	0.5256	3.5486	2.8671
Std. Dev.	0.0219	0.0207	0.1891	0.1592

demonstrate that HCTN remains computationally lightweight while providing strong predictive capability, making it suitable for large-scale, time-aware QoS prediction scenarios where both accuracy and deployability are essential.

5.2.2.6 Statistical Analysis

Table 5.8 shows the statistical analysis of HCTN across all datasets, using three different confidence limits (CL): 90%, 95%, and 99%. For example, a CL of 90% indicates that the given samples of QoS values are 90% likely to contain the population's true mean. The confidence interval analysis shows that as the CL increases, the range of the interval broadens, indicating greater certainty about capturing the true mean of QoS values, but with a wider range. This reflects a trade-off between precision and confidence in the performance estimates of HCTN.

Table 5.9: HCTN Inference time on different machines

Processor Configuration	Core	RAM	Inference Time (second) (mean ± stdev)
10th Gen Intel® Core™ i7-10700@4.80 GHz	16	7.53 GB	2.58e-5 ± 1.19e-6
13th Gen Intel® Core™ i5-1335U@4.60 GHz	12	15.30 GB	1.67e-5 ± 1.66e-6
AMD Ryzen-9 7950X@5.70 GHz	16	32.00 GB	2.12e-6 ± 6.24e-8
13th Gen Intel® Core™ i9-13900H@5.40 GHz	20	30.98 GB	2.01e-5 ± 2.30e-6
12th Gen Intel® Core™ i7-12700@4.90 GHz	20	62.49 GB	1.06e-5 ± 1.64e-7
12th Gen Intel® Core™ i7-12700@4.90 GHz	20	125.49 GB	1.10e-5 ± 1.35e-7
2nd Intel® Xeon Gold 6226R CPU@3.90GHz	32	188.35 GB	2.43e-5 ± 1.05e-7

5.2.3 Model Deployability

We analyze the deployability of HCTN by measuring its inference latency across a diverse set of hardware configurations, ranging from mid-tier consumer CPUs to high-end workstation processors. As shown in Table 5.9, HCTN consistently achieved low inference times, with mean latency ranging from 2.12×10^{-6} second to 2.58×10^{-5} second across all machines. These results demonstrate that HCTN introduces negligible computational overhead and delivers real-time predictions without requiring GPU acceleration. The low variance in latency further indicates stable execution behavior across different memory capacities and core counts. Overall, the study confirms that HCTN is an efficient, lightweight, and easily deployable in practical QoS-aware service platforms under resource-constrained environments.

In summary, HCTN demonstrates superior performance compared to previous methods, with significant improvements in MAE and RMSE across various datasets and training densities. It also excels in handling outliers, provides efficient training and prediction times, and benefits from advanced feature integration, including higher-order graph features and multi-granularity temporal features. A more detailed experimental analysis, including the impact of cold-start and the influence of various hyperparameters on the performance of HCTN, is presented in Appendices F and G, respectively.

5.3 Summary

This chapter introduces HCTN, a Hypergraph Convoluted Transformer Network, for anomaly-resilient real-time temporal QoS prediction. To address data deficiency, HCTN combines non-negative matrix decomposition with hypergraph-based collaborative filtering for richer, higher-order features. It handles data credibility by using a robust loss and a grey-sheep-aware time-sensitive strategy. For effective representation learning, HCTN integrates domain-specific and deep hypergraph-transformer-based features. The model outperformed existing methods in accuracy, with negligible inference latency, making it apt for real-time systems. In the future, we will explore

energy-efficient ML service selection for sustainable deployment.

5.4 Limitations

HCTN presents an advanced time-aware QoS prediction framework that improves robustness against sparsity, anomalies, and dynamic interaction patterns by learning credibility-aware and higher-order collaborative representations. However, several incremental challenges remain:

(i) Although HCTN leverages hypergraph convolution to capture higher-order relationships, it does not fully exploit hierarchical structural dependencies inherent in QoS data, such as nested relationships arising from geographical regions, network topology, and service-level groupings. Consequently, certain structural correlations may not be optimally represented.

(ii) HCTN focuses on single-attribute QoS prediction and does not support joint modeling of multiple QoS parameters. This limits its ability to capture interdependencies among QoS attributes and increases computational overhead when separate models are required for each QoS parameter.

Chapter 6

Sparsely-gated Hierarchical Adaptive Routing for joint Prediction of QoS

Existing methods in the literature rely on a single metric for QoS prediction. More recent methods adopt multi-task approaches to simultaneously predict multiple QoS parameters. However, these models suffer from the task-dominance problem during learning due to differences in the numerical ranges of QoS parameters, leading to negative transfer. Furthermore, relying on Euclidean representation learning, it captures limited hierarchical dependencies, which constrain the effectiveness of joint QoS modeling under sparse and noisy conditions. Despite these advances, existing methods still overlook implicit hierarchical dependencies and shared contextual correlations across QoS parameters. Moreover, task-balancing strategies used in works such as [123, 125] often exhibit numerical instability and oscillatory convergence, resulting in suboptimal multi-task performance. To address these limitations, we propose a unified framework for joint QoS prediction, called SHARP-QoS, which allows adaptive multi-task rep-

This chapter is derived from:

- **Suraj Kumar**, Arvind Kumar, Soumi Chattopadhyay, "SHARP-QoS: Sparsely-gated Hierarchical Adaptive Routing for Joint Prediction of QoS", IEEE Transactions on Services Computing. https://arxiv.org/pdf/2512.17262. (Under Review)

resentation learning by leveraging contextual data and complementary QoS signals. In this chapter, we introduce **S**parsely-gated **H**ierarchical **A**daptive **R**outing for joint **P**rediction of **QoS** (SHARP-QoS), which integrates non-negative MF–based QoS features with contextual information derived from AS and region, ensuring domain-aware and privacy-preserving initial representations. To capture the implicit hierarchical features across both QoS and contextual domains, we employ hyperbolic convolutional networks. Leveraging subnetwork-level routing, the framework enables adaptive feature sharing for QoS and contextual features, followed by a gated fusion module that dynamically selects between hierarchical QoS representations and shared features. To ensure reliable joint learning, we incorporate a robust loss function and an EMA-based loss-balancing strategy to mitigate negative transfer. Collectively, the proposed framework substantially enhances joint QoS prediction performance while remaining scalable, sparsity-tolerant, and resilient to noise.

More specifically, SHARP-QoS framework comprises three key components: (i) We introduce a Hierarchical Feature Extraction Block (HFEB) to capture implicit hierarchical dependencies across QoS and contextual features, enabling richer representations while preserving user and service privacy. (ii) Inspired by Sub-Network Routing (SNR) [166], a dual feature exchange mechanism is designed to share across both QoS- and contextual-features, while maintaining computational efficiency, unlike previous approaches. Additionally, a gated feature fusion module is employed to selectively integrate relevant representations from shared and structure-aware features. (iii) The Exponential Moving Average (EMA) is introduced to smooth short-term fluctuations in training due to loss scale, ensuring robust optimization for joint QoS prediction. The key contributions of this chapter are summarized below:

(i) **Novel Solution Architecture**: We propose a data-driven joint QoS prediction framework that leverages implicit hierarchical features from both QoS and context data. A feature-sharing and fusion layer is introduced that enables effective parameter sharing through QoS and contextual features, and dynamically selects task-specific features among structural and shared representations, resulting in enhanced QoS prediction performance.

(ii) **Handling Negative Transfer**: We introduce an EMA-based loss balancing strategy that reduces short-term oscillation and resolves inter-attribute conflicts across multiple QoS tasks, mitigating the negative transfer problem caused by numerical range discrepancies among QoS parameters, leading to faster convergence and improved generalization across tasks.

(iii) **Extensive experiments**: We conducted comprehensive evaluation of SHARP-QoS on three datasets: WSDREAM-1 [1], Reliability [123], and a gRPC dataset [167], covering two, three, and four QoS parameters, respectively. To ensure the reliability of our results, we include ablation studies, hyperparameter analysis, impact of outliers and cold-start, and statistical significance tests, demonstrating the effectiveness of our method.

The rest of this chapter is organized as follows. Section 6.1 presents a preliminary background on Hyperbolic geometry. Section 6.2 present proposed solution. The experiments are analyzed in Section 6.3. Finally, Section 6.4 concludes this chapter.

6.1 Preliminaries

Hyperbolic geometry has emerged as a powerful tool for modeling hierarchical and scale-free structures due to its constant negative curvature and exponential expansion [168]. In contrast to Euclidean space (zero curvature), hyperbolic space represents hierarchies and heavy-tailed degree distributions with lower distortion [169]. We adopt the Poincaré ball model to perform graph convolution in hyperbolic space, as it provides a conformal manifold representation and supports Riemannian optimization. Following [168, 169], we summarize the key definitions used in this work.

6.1.1 Poincaré Ball Model

The d-dimensional Poincaré ball with curvature $c{>}0$ is

$$\mathbb{B}^{d,c} = \{\mathbf{x}{\in}\mathbb{R}^d \mid c\|\mathbf{x}\|^2 < 1\},$$

equipped with the conformal metric

$$g_{\mathbf{x}} = \lambda_{\mathbf{x}}^2 g_{\mathrm{E}}, \quad \lambda_{\mathbf{x}} = 2/(1 - c\|\mathbf{x}\|^2),$$

and g_{E} denotes the standard Euclidean inner product. The conformal factor $\lambda_{\mathbf{x}}$ controls geometric distortion and is central to defining exponential and logarithmic maps. We use a learnable curvature, parameterized as $c = \mathrm{softplus}(r_c) + \epsilon$, with trainable r_c and a small $\epsilon > 0$, ensuring that the curvature remains strictly positive during training.

6.1.2 Hyperbolic Operations at the Origin

All computations are performed in the tangent space at the origin $< \mathbf{0} >$ for numerical stability [168].

(i) Exponential and logarithmic map: The exponential map projects the tangent vector v onto the hyperbolic manifold, while the logarithmic map performs the inverse operation by lifting hyperbolic points to the Euclidean space.

$$\begin{aligned} \exp^c_{<0>}(v) &= \tanh(\sqrt{c}\|v\|)\frac{v}{\sqrt{c}\|v\|}, \\ \log^c_{<0>}(x) &= \frac{1}{\sqrt{c}}\,\mathrm{artanh}(\sqrt{c}\|x\|)\frac{x}{\|x\|} \end{aligned} \tag{6.1}$$

(ii) Möbius addition ($\oplus_c$): This defines the gyrovector structure supporting hyperbolic vector operations.

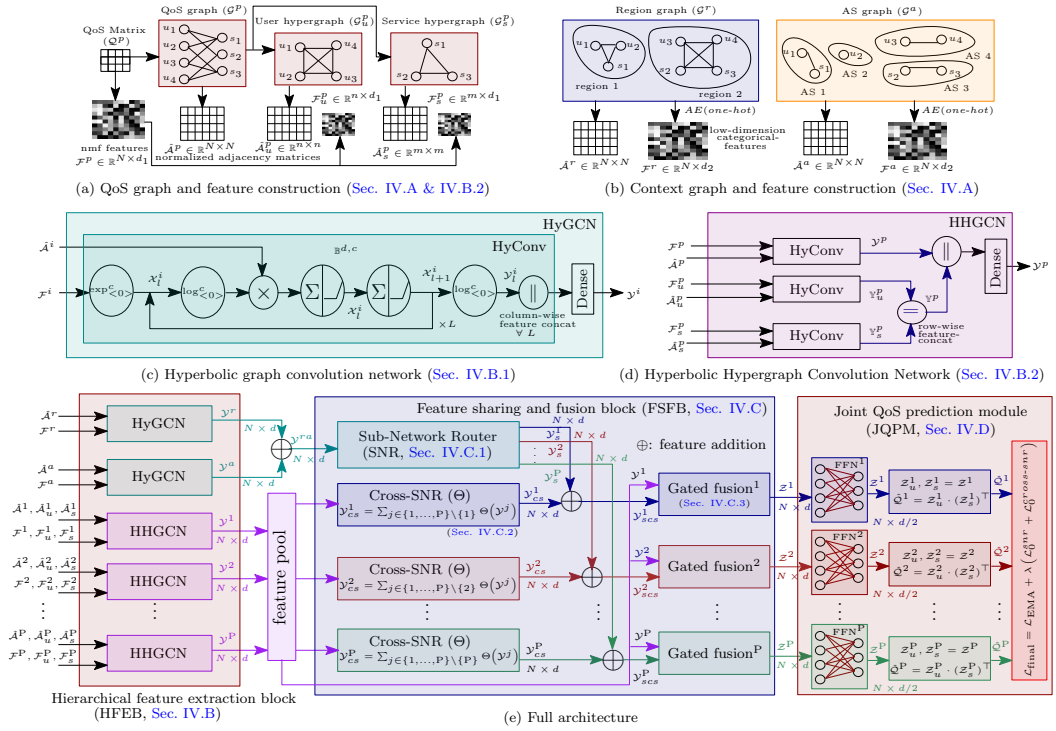
$$x \oplus_c y = \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y\rangle + c^2\|x\|^2\|y\|^2}. \tag{6.2}$$

(iii) Möbius matrix–vector multiplication ($\otimes_c$): This enables linear transformations in hyperbolic space.

$$W \otimes_c x = \exp^c_{<0>}(W\ \log^c_{<0>}(x)). \tag{6.3}$$

(iv) Wrapped activation ($\sigma^{\otimes_c}$): This applies nonlinear function (σ) while keeping outputs on the manifold.

$$\sigma^{\otimes_c}(x) = \exp^c_{<0>}(\sigma(\log^c_{<0>}(x))). \tag{6.4}$$



(a) QoS graph and feature construction (Sec. IV.A & IV.B.2)

(b) Context graph and feature construction (Sec. IV.A)

(c) Hyperbolic graph convolution network (Sec. IV.B.1)

(d) Hyperbolic Hypergraph Convolution Network (Sec. IV.B.2)

(e) Full architecture

Figure 6.1: SHARP-QoS: Overall framework.

6.2 Methodology

Following the classical multi-QoS prediction formulation defined in Definition 7, SHARP-QoS aims to address the following research questions: (i) how to effectively learn hierarchical representations of QoS and contextual entities that capture structural and latent dependencies, (ii) how to enable adaptive and controlled feature sharing across multiple QoS parameters while preserving task-specific characteristics, (iii) how to jointly leverage QoS interactions and contextual information to learn enriched multi-context representations, and (iv) how to achieve stable and balanced joint QoS prediction while mitigating negative transfer across QoS parameters. To address these challenges, SHARP-QoS introduces a unified framework that integrates hyperbolic graph-based hierarchical representation learning, adaptive subnetwork routing-based feature sharing and fusion, and EMA-balanced joint optimization for accurate and reliable multi-QoS prediction.

Fig. 6.1 illustrates the SHARP-QoS framework, comprising four key stages: *(i)*

Preprocessing, which constructs the QoS and context graphs and generates initial node features via matrix factorization and one-hot encoding; *(ii) Hierarchical Feature Extraction Block (HFEB)*, learning QoS and context specific representations using hyperbolic graph convolution; *(iii) Feature Sharing and Fusion Block (FSFB)*, which adaptively fuses QoS-based and contextual features across QoS parameters using a subnetwork-routing strategy and gated fusion; *(iv) Joint QoS Prediction Module (JQPM)*, to achieve joint QoS prediction via a feed-forward network (FFN), and matrix product, while training under EMA-based loss balancing to mitigate negative transfer.

Note that all hyperbolic computations follow the Poincaré ball formulation in Sec. 6.1, whereas feature sharing, fusion, and QoS prediction operate in Euclidean space.

6.2.1 Preprocessing

We first build the propagation graphs and initialize node features from the QoS matrices and context attributes.

6.2.1.1 QoS- and Context-Graph Construction

Learning on graphs can substantially improve QoS prediction; however, prior methods either relied on fully connected structures or used QoS/context graphs [55] with noisy or uninformative connections, limiting representation learning. To address this, our framework employs both QoS-invocation graphs and context graphs, defined as follows.

Definition 23 (QoS Invocation Graph ($\mathcal{G}^p$))**.** *Given a QoS matrix $\mathcal{Q}^p$, $p \in \{1, \ldots, \mathrm{P}\}$, the invocation graph $\mathcal{G}^p = \{\mathcal{V}, \mathcal{E}^p\}$ is a bipartite graph, where an edge $(u_i, s_j) \in \mathcal{E}^p$ exists iff $\mathcal{Q}^p_{ij} > 0$, indicating that user u_i invoked service s_j.*

Definition 24 (Context Graphs ($\mathcal{G}^r, \mathcal{G}^a$))**.** *For each context attribute $i \in \{r, a\}$, we define $\mathcal{G}^i = (\mathcal{V}, \mathcal{E}^i)$, where edges connect entities sharing the same region (r) or AS (a) attribute.*

The graph $\mathcal{G}^p$ captures collaborative QoS usage patterns, while $\mathcal{G}^r$ encodes coarse geographical proximity, and $\mathcal{G}^a$ captures fine-grained infrastructural priors (e.g., routing and peering). Although $\mathcal{G}^r$ may subsume many connections in $\mathcal{G}^a$, the two provide complementary geographic and network-structural perspectives. Empirically, combining $\{\mathcal{G}^p, \mathcal{G}^r, \mathcal{G}^a\}$ enhances representation learning while preserving user–service privacy, as only global contextual attributes are used (excluding sensitive information such as IP, GPS coordinates, or provider identity).

We represent each graph using its adjacency matrix $\mathcal{A}^i$, $i \in \{r, a, \{1, \ldots, \mathrm{P}\}\}$, producing P+2 matrices of size $N \times N$, where $N = n + m$. Directly using these matrices in GCNs [67] may cause numerical instability due to high-degree nodes and missing self-information. Thus, we adopt symmetrically normalized adjacency matrices:

$$\bar{\mathcal{A}}^i = (\mathcal{D}^i)^{-1/2} \cdot (\mathcal{A}^i + \mathcal{I}) \cdot (\mathcal{D}^i)^{-1/2}, \qquad \mathcal{D}^i_{jj} = \sum_k \mathcal{A}^i_{jk}, \tag{6.5}$$

where $\mathcal{D}^i$ is the degree matrix and $\mathcal{I}$ adds self-loops to preserve node self-features.

6.2.1.2 Initial Features Extraction

We derive initial node features using the sparse QoS matrices and the contextual attributes (RG, AS).

(i) QoS-based features: To obtain QoS features for each user and service node, we apply nonnegative matrix factorization [32] to each QoS parameter p, as shown in Eq. 6.6, yielding two low-rank matrices: the user feature matrix $\mathcal{F}^p_u \in \mathbb{R}^{n \times d_1}$ and the service feature matrix $\mathcal{F}^p_s \in \mathbb{R}^{m \times d_1}$.

$$\mathcal{F}^p_u,\ \mathcal{F}^p_s \leftarrow \mathcal{Q}^p \quad \text{such that} \quad \mathcal{F}^p_u \cdot (\mathcal{F}^p_s)^\top \approx \mathcal{Q}^p \tag{6.6}$$

(ii) Context-based features: To generate contextual embeddings for each user and service node, we employ one-hot encoding on RG and AS categories. Since one-hot vectors scale linearly with the number of categories and thus become high-dimensional, we use four autoencoders (AE) [133] with an identical feature dimension d_2 to obtain

compact representations. This produces four contextual feature matrices: user features $\mathcal{F}_u^r$, $\mathcal{F}_u^a \in \mathbb{R}^{n \times d_2}$ and service features $\mathcal{F}_s^r$, $\mathcal{F}_s^a \in \mathbb{R}^{m \times d_2}$ corresponding to RG and AS contexts.

We then form the combined node feature matrix for each source $i \in \{r, a, \{1, \ldots, \mathrm{P}\}\}$ by concatenating the user and service features row-wise, yielding $\mathcal{F}^i \in \mathbb{R}^{N \times d}$.

6.2.2 Hierarchical Feature Extraction Block (HFEB)

Previous approaches [55, 117, 122] employing graph convolution or attention networks (GCN/GAT) aggregate information from neighboring nodes in Euclidean space to extract higher-order features for QoS prediction. However, Euclidean aggregation fails to capture the inherent hierarchical structure present in QoS and contextual data, leading to high distortion and suboptimal representations. To address this limitation, we adopt the hyperbolic formulations described in Sec. 6.1, enabling graph representation learning in the Poincaré ball $\mathbb{B}^{d,c}$, where feature transformations and nonlinearities are realized via Möbius operations. Empirically, this yields more expressive representations by exploiting the underlying hierarchical structure, outperforming prior Euclidean approaches.

Specifically, we propose a dual hierarchical feature extraction mechanism, which is separately employed on the QoS and context graphs, as detailed subsequently.

6.2.2.1 Hyperbolic Graph Convolution Network (HyGCN)

Following [168, 169] (summarized in Sec. 6.1), we adopt a Hyperbolic Graph Convolution Network (HyGCN) to extract hierarchical features from the context graphs $\mathcal{G}^r$ and $\mathcal{G}^a$.

Given an initial feature matrix $\mathcal{F}^i$ and normalized adjacency $\bar{\mathcal{A}}^i$, where $i \in \{r, a\}$, HyGCN employs hyperbolic convolution (HyConv) unit that comprises two core operations, as depicted in Fig. 6.1 (c). Before using HyConv operations, we project the initial features to the hyperbolic manifold, $\mathcal{X}_0^i = \exp_{<0>}^c(\mathcal{F}^i)$. Following this, it first applies hyperbolic graph convolution, executing Euclidean message passing [67] in the

tangent space. Secondly, a hyperbolic nonlinear transformation is performed through Möbius linear mapping and wrapped activation with ReLU. The process is defined in Eq. 6.8.

$$\mathcal{X}_l^i = \sigma^{\otimes_c}\left(\exp_{<0>}^c\left(\bar{\mathcal{A}}^i \log_{<0>}^c(\mathcal{X}_l^i)\,\mathcal{W}_{l1}^i\right) \oplus_c \exp_{<0>}^c(b_{l1})\right) \tag{6.7}$$

$$\mathcal{X}_{(l+1)}^i = \sigma^{\otimes_c}\left(\exp_{<0>}^c\left(\log_{<0>}^c(\mathcal{X}_l^i)\,\mathcal{W}_{l2}^i\right) \oplus_c \exp_{<0>}^c(b_{l2})\right). \tag{6.8}$$

where, $\mathcal{W}_{l1}^i \in \mathbb{R}^{d\times 2d}$ and $\mathcal{W}_{l2}^i \in \mathbb{R}^{d\times d}$ are learnable weights, with corresponding biases $b_{l1} \in \mathbb{R}^{2d}$ and $b_{l2} \in \mathbb{R}^d$. We then return to Euclidean space via $\mathcal{Y}_l^i = \log_{<0>}^c(\mathcal{X}_l^i) \in \mathbb{R}^{N\times d}$.

HyGCN stack L HyConv layers whose outputs are concatenated column-wise, and then a nonlinear transformation layer is applied with learnable weight matrix $\mathcal{W}_1^i \in \mathbb{R}^{d(l+1)\times d}$ with an activation function σ_1, as shown in Eq. 6.9, to obtain the higher-order hierarchical features. Note that this operation is performed entirely in Euclidean space.

$$\mathcal{Y}^i = \sigma_1\left(\|_{l=0}^L \mathcal{Y}_l^i\right) \cdot \mathcal{W}_1^i \tag{6.9}$$

We independently employ HyGCN over $\mathcal{G}^r$ and $\mathcal{G}^a$, yielding hierarchical contextual features $\mathcal{Y}^r \in \mathbb{R}^{N\times d}$ and $\mathcal{Y}^a \in \mathbb{R}^{N\times d}$. Adding $\mathcal{Y}^{ra} = \mathcal{Y}^r + \mathcal{Y}^a$ provides the aggregated region and network-aware, structural contextual features, which we used in Sec. 6.2.3.

6.2.2.2 Hyperbolic Hypergraph Convolution Network (HHGCN)

Adopting HyGCN directly for QoS-based graphs $\mathcal{G}^p$ for each QoS parameter p, may result in suboptimal QoS-based hierarchical features due to weak structural information owing to its sparse or noisy QoS interactions. We therefore augment the bi-partite structural signals using user- and service-based hypergraphs. The user-based hypergraph $\mathbb{G}_u^p$ is defined as follows:

Definition 25 (User Hypergraph ($\mathbb{G}_u^p$))**.** *Given a QoS invocation graph $\mathcal{G}^p$, we derive user (service) hypergraph defined as $\mathbb{G}_u^p = \{\mathbb{V}_u^k, \mathbb{E}_u^k\}$ where $\mathbb{V}_u^k \in \mathcal{U}$ and $\mathbb{E}_u^k = \{e_1, e_2, \ldots, e_m\}$ are user set and hyperedge set, respectively.*

The service-based hypergraph $\mathbb{G}_s^p$ is obtained in a similar manner. Notably, $\mathbb{G}_u^p$ and $\mathbb{G}_s^p$ are constructed via a second-hop traversal on $\mathcal{G}^p$ with incidence matrix $\mathcal{H}^p$.

Consequently, we obtain the normalized adjacency matrices $\bar{\mathbb{A}}_u^p \in \mathbb{R}^{n\times n}$ and $\bar{\mathbb{A}}_s^p \in \mathbb{R}^{m\times m}$ for each graphs $\mathbb{G}_u^p$ and $\mathbb{G}_s^p$, respectively, as shown in Eqs. 6.10 and 6.11.

$$\bar{\mathbb{A}}_u^p = (\mathbb{D}_u^p)^{-1/2} \cdot \mathcal{H}^p \cdot (\mathbb{D}_s^p)^{-1} \cdot (\mathcal{H}^p)^\top \cdot (\mathbb{D}_u^p)^{-1/2}, \tag{6.10}$$

$$\bar{\mathbb{A}}_s^p = (\mathbb{D}_s^p)^{-1/2} \cdot (\mathcal{H}^p)^\top \cdot (\mathbb{D}_u^p)^{-1} \cdot \mathcal{H}^p \cdot (\mathbb{D}_s^p)^{-1/2}. \tag{6.11}$$

Here, $\mathbb{D}_u^p$ and $\mathbb{D}_s^p$ are degree matrices obtained using the $\mathbb{A}_u^p$ and $\mathbb{A}_s^p$ via hypergraphs $\mathbb{G}_u^p$ and $\mathbb{G}_s^p$, respectively.

We employ Hyperbolic Hypergraph Convolution Network (HHGCN), shown in Fig. 6.1 (d), leveraging the HyConv formulations (see Eq. 6.8–6.9) to each QoS graphs/hypergraphs $\mathcal{G}^p, \mathbb{G}_u^p$ and $\mathbb{G}_s^p$ independently, and resulting outputs are then bring to Euclidean space producing $\mathcal{Y}_l^p$, $\mathbb{Y}_{ul}^p$ and $\mathbb{Y}_{sl}^p$, respectively. $\mathbb{Y}_{ul}^p$ and $\mathbb{Y}_{sl}^p$ are first concatenated in a row-wise manner, whose output is then concatenated column-wise with $\mathcal{Y}_l^p$. The final consolidated features are obtained by employing a non-linear transformation with learnable weights $\mathcal{W}_2^p \in \mathbb{R}^{2d(l+1)\times d}$ and an activation function σ_1, as shown in Eq. 6.12.

$$\mathcal{Y}^p = \sigma_1\big(\|_{l=0}^{L} (\mathcal{Y}_l^p \parallel [\mathbb{Y}_{ul}^p ; \mathbb{Y}_{sl}^p]) \cdot \mathcal{W}_2^p\big) \tag{6.12}$$

The resulting output $\mathcal{Y}^p \in \mathbb{R}^{N\times d}$ encodes QoS-based hierarchical features for p-th QoS parameter, which is utilized in the next section.

6.2.3 Feature Sharing and Fusion Block (FSFB)

This section presents our approach to controlled feature sharing based on representations learned via HyGCN and HHGCN for joint QoS prediction. Since feature sharing is an algorithmic design choice rather than geometric modeling, all operations in this stage are performed in Euclidean space.

Existing joint QoS prediction methods [92, 117, 121–123, 125] suffer from limited flexibility in parameter sharing or computational overhead. Hard parameter sharing forces all tasks to share early layers, often causing negative transfer, while soft parameter sharing relies on task-specific parameters with similarity regularization, increasing model complexity and limiting scalability. To address these issues, inspired by [166],

we introduce a subnetwork routing strategy that enables adaptive, task-specific feature sharing through selective routing over shared subnetworks.

6.2.3.1 Subnetwork Routing (SNR) Mechanism

In our proposed SHARP-QoS framework, QoS and context features are shared in distinct ways. We first explain the Sub-Network Routing Mechanism (SNR) [166] for contextual features $\mathcal{Y}^{ra} \in \mathbb{R}^{N\times d}$ obtained via HyGCNs (Sec. 6.2.2.1). We dynamically select and aggregate a sparse subset of transformed features conditioned on the target task. To achieve this, SNR router first process $\mathcal{Y}^{ra}$ through K_1 parallel blocks $\{\phi_k(\cdot)\}_{k=1}^{K_1}$, each comprise layer normalization (LN), non-linear transformation with ReLU activation (Dense), as shown in Eq. 6.13.

$$\phi_k(\mathcal{Y}^{ra}) = \text{Dense}_k(\text{LN}(\mathcal{Y}^{ra}))\,, \qquad \phi_k(\mathcal{Y}^{ra}) \in \mathbb{R}^{N\times d_o} \tag{6.13}$$

where d_o is the output dimension. We further employ a QoS parameter specific linear transformation with $\mathcal{W}_k^p$, helping to enhance the task specificity, as shown in Eq. 6.14.

$$\tilde{\phi}_k^p(\mathcal{Y}^{ra}) = \phi_k(\mathcal{Y}^{ra}) \cdot \mathcal{W}_k^p, \tag{6.14}$$

where $\mathcal{W}_k^p \in \mathbb{R}^{d_o\times d_o}$ is learned independently. To enable adaptive feature sharing for each QoS parameter, we maintain learnable coding variables $\{c_k^p\}_{p=1}^{\text{P}}$ per block, controlling the connection. Specifically, we draw each c_k^p from a Bernoulli distribution π_k^p, owing to its discrete nature, it does not allow efficient gradient-based learning. To address this issue, we leverage hard concrete distribution [170], where each coding variable c_k^p parameterized by a learnable logit $\log \alpha_k^p$, sampled using a uniform distribution $u \sim \mathcal{U}(0,1)$.

$$s = \sigma_2\left(\frac{(\log u - \log(1-u) + \log \alpha_k^p)}{\tau}\right), \tag{6.15}$$

$$\bar{s} = s(\gamma - \beta) + \beta, \tag{6.16}$$

$$c_k^p = \text{clip}(\bar{s}, 0, 1). \tag{6.17}$$

where, σ_2 is an activation function, $\beta < 0, \gamma > 1$ define the stretching interval, and τ controls the smoothness. Subsequently, we obtain the QoS parameter-specific features

by using a sparsely weighted aggregation using the coding variable c_k^P, as illustrates in Eq. 6.18.

$$\mathcal{Y}_s^p = \frac{1}{\sum_{k=1}^{K} g_k^p} \sum_{k=1}^{K_1} c_k^p \, \tilde{\phi}_k^p(\mathcal{Y}^{ra}), \tag{6.18}$$

where the normalization term helps prevent scale collapse when only a small number of blocks are activated, allowing fractional routing in training. During inference, we follow the hard selection where stochastic coding variables are replaced by deterministic binary activations by thresholding (δ),

$$c_k^p = \mathbb{I}[\sigma_2(\log \alpha_k^p) > \delta] \tag{6.19}$$

yielding interpretable and task-specialized contextual features.

6.2.3.2 Cross-SNR Routing Mechanism

Since QoS parameters stem from common underlying network conditions, we argue that they either depend on or influence each other. However, at the same time, they may not have trended all the time. Previous studies have undermined these challenges and avoided sharing QoS-based features. Addressing this issue, we introduced cross-subnetwork routing (Cross-SNR) strategy to obtain the task-specialized QoS features, exploiting the shareable component within the QoS-based features for other QoS parameters. Cross-SNR $\Theta(\cdot)$ employs the same formulations, as in Eqs. 6.13-6.18 with K_2 number of blocks, on QoS features pool $\{\mathcal{Y}^p\}_{p=1}^{\mathrm{P}}$ derived using HHGCNs (Eq. 6.12). For each target QoS parameter p, we construct a shared representation by excluding its own feature and summing the remaining $\mathrm{P}-1$ task-specialized QoS features outputs, as shown in Eq. 6.20.

$$\mathcal{Y}_{cs}^p = \sum_{j \in \{1,\ldots,\mathrm{P}\} \setminus \{p\}} \Theta\big(\mathcal{Y}^j\big) \tag{6.20}$$

We further obtain the combined shared features $\mathcal{Y}_{scs}^p = \mathcal{Y}_s^p + \mathcal{Y}_{cs}^p$ by summing task-specialized contextual and QoS features, yielding multi-context shared features, enriched in the network, geographical, and QoS dynamics.

6.2.3.3 Gated-feature Fusion Module

This module introduces the adaptive strategy to combine the shared task-specific representations $\mathcal{Y}^p_{scs} \in \mathbb{R}^{N \times d}$ with the structure-aware QoS features $\mathcal{Y}^p \in \mathbb{R}^{N \times d}$ (computed in Eq. 6.12, Fig. 6.1 (e)). Specifically, we introduce a gated feature fusion mechanism, which learn a gating weight $\mathcal{W}^p_g$ with an activation function σ_2 producing the resulting gates g^p for each QoS parameter p, as shown in Eq. 6.21. The gates g^p enable the model to adaptively balance structural and multi-context shared information, illustrated in Eq. 6.22, where $\odot$ represents the element-wise multiplication. The resulting feature $\mathcal{Z}^p$ is then forwarded for the downstream QoS prediction task.

$$g^p = \sigma_2\left([\mathcal{Y}^p || \mathcal{Y}^p_{scs}] \cdot \mathcal{W}^p_g\right) \tag{6.21}$$

$$\mathcal{Z}^p = g^p \odot \mathcal{Y}^p + (1 - g^p) \odot \mathcal{Y}^p_{scs} \tag{6.22}$$

6.2.4 Joint QoS Prediction Module (JQPM)

To enable the joint learning, we obtain task-specific feed-forward networks $\text{FFN}^p(\cdot)$ for each QoS parameter p, which comprises two dense layers with ReLU and Linear activation, respectively. Subsequently, the resulting features $\mathcal{Z}^p$ is split into QoS parameter specific user feature matrix $\mathcal{Z}^p_u$ and service feature matrix $\mathcal{Z}^p_s$. We employ the matrix multiplication on $\mathcal{Z}^p_u$ and $\mathcal{Z}^p_s$, providing the predicted QoS matrix $\hat{\mathcal{Q}}^p$. This formulation is illustrates in Eqs. 6.23-6.25.

$$\mathcal{Z}^p = \text{FFN}^p(\mathcal{Z}^p), \tag{6.23}$$

$$\mathcal{Z}^p_u, \mathcal{Z}^p_s = \mathcal{Z}^p \tag{6.24}$$

$$\hat{\mathcal{Q}}^p = \mathcal{Z}^p_u \cdot (\mathcal{Z}^p_s)^\top \tag{6.25}$$

6.2.4.1 Objective Function and Loss Scale Balancing

Owing to network instability, QoS data may contain outliers; therefore, we train our model using a robust loss function, Mean Absolute Error (MAE), illustrated in Eq. 6.26.

$$\mathcal{L}^p = \frac{1}{|\text{TrD}|} \sum_{(i,j) \in \text{TrD}} \left| \mathcal{Q}^p_{ij} - \hat{\mathcal{Q}}^p_{ij} \right| \tag{6.26}$$

where TrD denotes the training density, ensuring only observed QoS values contribute to the loss calculation.

We argue that the combined loss $\sum_{p=1}^{\mathrm{P}} \mathcal{L}^p$ is prone to being dominated by a few QoS parameters that yield larger error magnitudes, which might lead to negative transfer across tasks. This problem can be attributed to the numerical ranges of QoS parameters (see Table 2.3). Recent studies PMT [123] and WAMTL [125] employed heteroscedastic uncertainty weighting (HUW) and Dynamic Weight Averaging (DWA), respectively, to rescale task losses. However, they tend to suffer from instability during the early training stages and exhibit slow adaptation to QoS dynamics. To address this issue, we introduce an EMA-based loss scaling strategy that moderates task contributions while suppressing short-term fluctuations.

Specifically, at i-th training iteration, we estimate the smoothed loss $\tilde{\mathcal{L}}_i^p$ using a smoothing coefficient $\beta \in [0, 1)$, controlling the moving average momentum. The normalized QoS parameter specific weights w^p are then computed by inverting the smoothed losses, as shown in Eq. 6.27. Here, $\tilde{\mathcal{L}}_0^p = 1$ and ϵ is a small constant for numerical stability.

$$\tilde{\mathcal{L}}_i^p = \beta\, \tilde{\mathcal{L}}_{(i-1)}^p + (1-\beta)\, \mathcal{L}_i^p \ , \quad w^p = \frac{(\tilde{\mathcal{L}}^p + \epsilon)^{-1}}{\sum_{p=1}^{\mathrm{P}} (\tilde{\mathcal{L}}^p + \epsilon)^{-1}}. \tag{6.27}$$

The resulting joint loss is expressed as $\mathcal{L}_{\mathrm{EMA}} = \sum_{p=1}^{\mathrm{P}} w^p\, \mathcal{L}^p$. We further introduce a $\mathcal{L}_0$-based regularization term, encouraging the sparsity in the routing block selection. The final joint objective function is illustrated in Eq. 6.28, utilizes to train our model. Note that the regularization coefficient λ controls the trade-off between QoS-specific prediction accuracy and the sparsity across feature routing networks.

$$\mathcal{L}_{\mathrm{final}} = \mathcal{L}_{\mathrm{EMA}} + \lambda\, (\mathcal{L}_0^{snr} + \mathcal{L}_0^{cross\text{-}snr}) \tag{6.28}$$

Overall, our framework ensures that EMA-based balancing demonstrates smooth QoS dynamics, stable, and adaptive weighting. Further, combining with the sparse routing regularizer achieves effective feature sharing across QoS parameters, leading to balanced multi-task optimization, with improved QoS prediction performance. The complexity analysis of SHARP-QoS is illustrated in Appendix A of supp. file.

6.3 Experimental Results

We implement our framework using TensorFlow 2.19.0 with Python 3.12.7. All experiments were conducted on an Ubuntu 24.04.2 LTS (Linux kernel 5.15.0-139-generic, x86_64) equipped with a 12th Gen Intel(R) Core(TM) i7-12700 CPU @ 1.42 GHz and 130 GiB RAM.

6.3.1 Experimental Setup

We employ three real-world web service datasets to evaluate SHARP-QoS: WSDREAM-1 [1], Reliability dataset [123], and a gRPC dataset, which is introduced in Chapter 4. These datasets enable evaluation under two-, three-, and four-attribute QoS prediction settings, respectively.

To simulate realistic sparse dataset scenarios, we randomly sample and remove a fixed percentage TrD $\in \{5, 10, 15, 20\}\%$ of the QoS records from each dataset, where TrD represents training density. The remaining TeD $= (100 - \text{TrD})\%$ is used for performance evaluation. To ensure statistical reliability, we conducted multiple runs

Table 6.1: Parameters configuration.

Attributes		Value
NMF/Autoencoder	d_1/d_2 dimension	128
HyGCN/HHGCN	No. of layers (L)	2
	$\mathcal{W}_{l1}^i$ or $\mathcal{W}_{l2}^i$, & $\mathcal{W}_1^i$ dimension	128×128, & 256×128
	Activation function (σ_1)	ReLU
SNR/ Cross-SNR	No. of Blocks (K_1/K_2)	4
	$\mathcal{W}_k^p$ dimension	64×128
	Activation function & threshold (σ_2)	sigmoid, 0.5
Gated Fusion	$\mathcal{W}_g^p$ dimension	32×256
	Activation function (σ_2)	sigmoid
FFN	$\mathcal{W}_1, \mathcal{W}_2$ dimension	128×128, 128×64
	Activation function	ReLU, Linear
Objective Function	EMA coefficient (β)	0.99
	Regularization coefficient (λ)	1×10^{-5}
Training Parameters	No. of Epoch, Patience	10000, 400
	Optimizer	AdamW
	Learning rate, Decay rate	1×10^{-3}, 1×10^{-4}

Table 6.2: Performance comparison with joint QoS prediction methods on WSDREAM-1 dataset

QoS Parameter	Method	MAE				RMSE			
		5	10	15	20	5	10	15	20
RT	JQSP [117]	0.5079	0.4406	0.4154	0.4072	1.9491	1.4445	1.3496	1.3181
	WAMTL [125]	0.4652	0.4097	0.3844	0.3663	1.3607	1.2800	1.2345	1.2017
	DNM [92]	0.4121	0.3621	0.3471	0.3214	1.3866	1.2670	1.2216	1.2070
	MGEN [121]	0.4115	0.3423	0.3289	0.3181	1.3579	1.2602	1.2189	1.1950
	SHARP-QoS	**0.3668**	**0.3243**	**0.3099**	**0.2930**	**1.3116**	**1.2450**	**1.1979**	**1.1516**
	PG(%)	10.86	5.25	5.78	7.89	3.41	1.21	1.72	3.63
TP	JQSP [117]	21.9919	17.9796	16.8000	14.3989	79.5323	54.0659	49.8281	49.0998
	WAMTL [125]	18.8521	15.3414	14.5306	13.7570	54.5032	46.7399	43.5842	41.5083
	DNM [92]	17.2980	14.0650	14.5436	14.1170	58.1629	50.3930	49.8868	46.9100
	MGEN [121]	15.4529	13.0833	12.6516	12.3941	50.7213	43.3581	43.2147	42.1618
	SHARP-QoS	**13.2402**	**11.4814**	**10.8035**	**10.4069**	**47.0426**	**41.7156**	**39.7392**	**38.7746**
	PG(%)	14.32	12.24	14.61	16.03	7.25	3.79	8.04	6.59

of model training and reported the average results.

Unless specified otherwise, we use the hyperparameters listed in Table 6.1.

6.3.2 Performance Analysis

6.3.2.1 Comparison with past methods

Among all available joint QoS prediction baselines, we evaluated SHARP-QoS against [92, 117, 121, 125] via error metrics (MAE, RMSE) and computational efficiency. Comparison with PMT [123], and HTG [122] is discarded as they require topology-level AS-links data, which is publicly unavailable. To further strengthen our analysis, we compare with five single-parameter QoS prediction methods.

(i) Comparison with multi-task methods on error metrics: Table 6.2, Table 6.3, and Table 6.4 report a comprehensive comparison of SHARP-QoS against state-of-the-art (SOTA) joint QoS prediction baselines on four training densities (5% – 20%, step size 5%) across two, three, and four QoS parameters, respectively. While prior methods demonstrate inconsistent behavior across settings, our approach delivers uniformly superior performance, achieving average improvements of 19.47% in MAE and 19.32% in RMSE over second-best models across all datasets. We observed that performance gains become even more pronounced as the number of tasks in-

Table 6.3: Performance comparison with joint QoS prediction methods on the Reliability dataset

QoS Parameter	Method	MAE				RMSE			
		5	10	15	20	5	10	15	20
RT	DNM [92]	1.1098	1.0413	1.0332	1.0016	2.4307	2.1130	2.0288	2.0040
	JQSP [117]	0.7688	0.5053	0.4495	0.4650	1.6266	1.3327	1.0606	1.1111
	MGEN [121]	0.6627	0.4216	0.3794	0.3639	1.4639	1.1895	1.0862	1.0790
	WAMTL [125]	0.6562	0.4771	0.4423	0.4093	1.2595	1.2268	1.1525	1.1269
	SHARP-QoS	**0.4496**	**0.3945**	**0.3099**	**0.2871**	**1.1652**	**1.1015**	**1.0263**	**1.0029**
	PG(%)	31.48	6.43	18.32	21.10	7.49	7.40	3.23	7.05
TP	JQSP [117]	54.2766	46.9502	50.9577	27.0942	253.2556	218.3714	112.5865	92.1144
	DNM [92]	43.2913	33.4560	35.6610	34.6610	116.1662	109.6389	106.0478	102.7562
	WAMTL [125]	33.6497	27.6057	23.2590	18.9285	132.5707	129.2284	126.6743	111.6510
	MGEN [121]	24.5219	18.1270	15.0493	14.9293	137.2764	107.0232	99.8948	78.2624
	SHARP-QoS	**24.4732**	**17.7958**	**14.4464**	**11.4730**	**88.3392**	**71.9799**	**72.4821**	**43.4623**
	PG(%)	0.20	1.83	4.01	23.15	23.95	32.74	27.44	44.47
RE	WAMTL [125]	0.2805	0.0941	0.0907	0.0699	0.3528	0.1383	0.1148	0.1094
	MGEN [121]	0.0484	0.0395	0.0164	0.0158	0.1926	0.1830	0.0972	0.0964
	JQSP [117]	0.0429	0.0348	0.0181	0.0116	0.0776	0.0662	0.0455	0.0426
	DNM [92]	0.0112	0.0109	0.0101	0.0091	0.0572	0.0547	0.0484	0.0459
	SHARP-QoS	**0.0100**	**0.0072**	**0.0062**	**0.0060**	**0.0344**	**0.0339**	**0.0326**	**0.0299**
	PG(%)	10.71	33.94	38.61	34.07	39.86	38.03	28.35	29.81

creases. Specifically, our method surpasses existing approaches on `WSDREAM-1` by 10.87% (MAE) and 4.46% (RMSE), on `Reliability dataset` by 18.65% (MAE) and 24.15% (RMSE), and on `gRPC-4T` by 28.88% (MAE) and 26.34% (RMSE). These results strongly highlight the scalability and generalization strength of our framework under multi-task expansion. Against MoE-based architectures such as MGEN [121] and WAMTL [125], our method secures substantial gains of 40.77% (MAE) and 40.99% (RMSE). We attribute this advantage to the stability of our design, in contrast to the load imbalance and expert collapse commonly observed in MoE-based training. JQSP, despite its graph-attention foundation, suffers from severe performance degradation, 47.66% in MAE and 35.91% in RMSE, due to its sensitivity to outliers. Likewise, DNM [92], though architecturally intricate, falls behind our method by 38.69% (MAE) and 29.43% (RMSE), reflecting its inability to mitigate negative transfer effectively.

Collectively, these results affirm the superiority of SHARP-QoS, demonstrating strong resilience to outliers, effective suppression of negative transfer, and consistent hierarchical feature extraction across all datasets.

Table 6.4: Performance comparison with joint QoS prediction methods on gRPC dataset

QoS Parameter	Method	MAE				RMSE			
		5	10	15	20	5	10	15	20
RE	WAMTL [125]	0.0475	0.0360	0.0290	0.0240	0.0688	0.0498	0.0423	0.0325
	JQSP [117]	0.0879	0.0753	0.0654	0.0626	0.1545	0.1329	0.1226	0.1139
	DNM [92]	0.0505	0.0345	0.0328	0.0310	0.0663	0.0495	0.0476	0.0449
	MGEN [121]	0.0422	0.0287	0.0253	0.0233	0.0706	0.0564	0.0440	0.0413
	SHARP-QoS	**0.0295**	**0.0200**	**0.0160**	**0.0123**	**0.0443**	**0.0475**	**0.0416**	**0.0286**
	PG(%)	30.09	30.31	36.76	47.21	33.18	4.04	5.45	30.75
CT	WAMTL [125]	1.1504	0.9515	0.7851	0.6334	1.4946	1.2406	1.0152	1.0023
	MGEN [121]	1.5543	1.1733	0.8303	0.7219	2.2099	1.5409	1.1259	0.9951
	DNM [92]	1.1464	0.8296	0.6902	0.6659	1.5566	1.1257	0.9239	0.8922
	JQSP [117]	1.0381	0.8232	0.7807	0.7852	1.3539	1.0784	1.0124	0.9981
	SHARP-QoS	**0.8836**	**0.7060**	**0.6008**	**0.5618**	**1.2044**	**0.9976**	**0.8452**	**0.8020**
	PG(%)	14.88	14.24	12.95	15.63	11.04	7.49	8.52	10.11
LT	JQSP [117]	4.8826	3.9492	3.6310	3.3331	9.4759	7.7057	7.5593	7.4431
	DNM [92]	4.3804	4.2163	3.4580	3.4164	9.2961	8.3004	7.9792	7.9022
	MGEN [121]	2.6290	2.0706	2.0523	2.0584	9.0301	8.6840	8.4091	5.6198
	WAMTL [125]	2.5137	3.5620	3.4255	3.6652	7.7507	10.4098	10.418	10.7406
	SHARP-QoS	**1.1910**	**0.8909**	**0.8241**	**0.7896**	**1.9780**	**1.7863**	**1.5810**	**1.5799**
	PG(%)	52.62	56.97	59.85	61.64	74.48	76.82	79.09	71.89
PW	WAMTL [125]	12.8474	9.2137	8.4201	7.3705	17.8271	16.3605	14.3976	12.9835
	JQSP [117]	13.8065	10.5663	10.0520	9.7656	17.7371	13.9167	13.2140	12.6584
	MGEN [121]	12.4266	8.0331	6.2375	5.6198	19.7486	14.4056	12.0813	11.1539
	DNM [92]	12.2556	9.0397	8.5686	8.2758	17.3730	14.3611	12.5894	12.1153
	SHARP-QoS	**11.3154**	**7.2178**	**6.2153**	**5.0122**	**16.8738**	**13.6852**	**12.0119**	**10.7687**
	PG(%)	7.67	10.15	0.36	10.81	2.87	1.66	0.57	3.45

(ii) Comparison with multi-task methods on computational complexity: To evaluate computational efficiency, we compare our framework across four metrics: trainable parameters (Params), floating-point operations (FLOPs), training time, and inference time, as shown in Table 6.5. While our model exhibits a slightly higher training time, it maintains moderate Params and FLOPs, and is the fastest among all baselines during inference. Importantly, training is performed offline; that is, the extended training time does not impact runtime service invocation or latency during deployment. Relative to WAMTL [125], which has the smallest parameter footprint (1.78× fewer than ours), our method delivers substantial gains with 5.83× lower FLOPs and an outstanding 1000× speedup in inference. Likewise, compared to JQSP [117], which reports the lowest FLOPs (ours being only 1.48× higher), our framework achieves 1.55× faster inference while also requiring 1.25× fewer param-

Table 6.5: Comparison on computational efficiency

Method	Params (M)	FLOPs (G)	Train Time (sec.)	Inference Time (sec.)
MGEN [121]	1.79	4774.56	3847.25	8.10×10^{-5}
WAMTL [125]	0.91	777.61	3901.07	1.35×10^{-5}
DNM [92]	2.44	211.8	2851.27	2.16×10^{-6}
JQSP [117]	2.02	90.12	2532.52	6.14×10^{-8}
SHARP-QoS	1.62	133.4	5516.00	3.97×10^{-8}

Table 6.6: Performance comparison with single-task methods on WSDREAM-1 [1].

Method	RT				TP			
	MAE		RMSE		MAE		RMSE	
	10	20	10	20	10	20	10	20
PMF [35]	0.4996	0.4492	1.2866	1.1828	16.1755	14.6694	46.4439	42.4855
CMF [42]	0.4511	0.3767	1.5012	1.3633	23.2347	18.8050	83.5279	76.5277
DCALF [113]	0.4544	0.4246	1.2450	1.1759	15.3595	13.6697	45.9013	41.2194
llmQoS [57]	0.3600	0.3270	1.2240	1.1590	12.0220	10.7600	42.9470	38.3650
QoSGNN [55]	0.3450	-	1.2760	-	13.9460	-	47.9550	-
SHARP-QoS	**0.3243**	**0.2930**	**1.2450**	**1.1516**	**11.4814**	**10.4069**	**41.7156**	**38.7746**
PG(%)	5.22	9.65	-1.72	0.64	4.50	3.28	2.87	-1.07

eters. Notably, despite lower compute, our approach consistently achieves better predictive accuracy across all datasets.

These results confirm that SHARP-QoS offers an advantageous trade-off between computational cost and performance, delivering high accuracy, reduced runtime overhead, and strong scalability, making it highly suitable for latency-critical and real-time QoS prediction systems.

(iii) Comparison with single-task methods: Table 6.6 presents the comparison of SHARP-QoS with five recent single-task methods on the `WSDREAM-1`. Our method consistently achieves the lowest MAE across all methods, demonstrating higher average prediction accuracy. However, in certain cases (RT-10, TP-20), our model achieves slightly higher RMSE, underperforms compared to the underlined method. This may be attributed to their dedicated optimization for individual QoS parameters.

Despite this, our approach provides a more nuanced solution by jointly optimizing multiple QoS parameters within a unified framework. This balance is highly relevant in real deployment scenarios, where service selection depends on multiple QoS properties rather than isolated metrics.

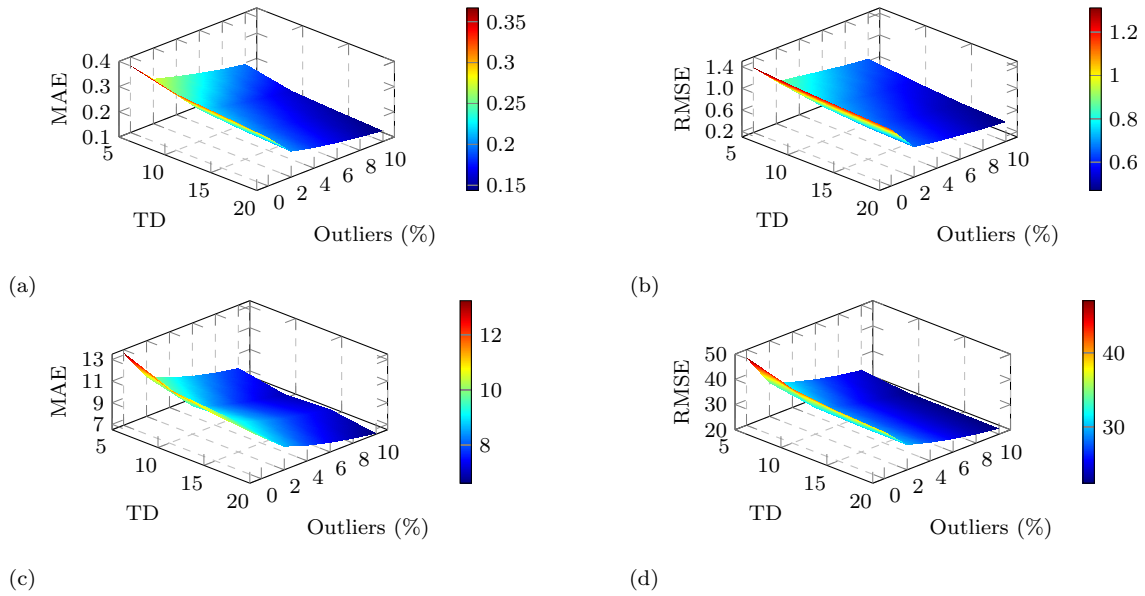


Figure 6.2: Impact of outliers on WSDREAM-2T dataset (a) RT (MAE), (b) RT (RMSE), (c) TP (MAE), and (d) TP (RMSE).

6.3.2.2 Outlier Sensitivity Analysis

Outliers have a substantial impact on QoS prediction performance [42]. We explicitly handle outliers during both training and inference. Empirical inspection shows that the WSDREAM-1 dataset contains a considerable number of anomalous QoS values. Thus, during training, we adopt the L_1 (MAE) loss, which is inherently more robust to outliers than the L_2 (MSE) loss. Furthermore, to quantify the influence of outliers during inference, we identify a fixed percentage of outliers using the i-Forest [109] and evaluate model performance after removing them.

Fig. 6.2 reports the results when eliminating 2%–10% of outliers (in increments of 2%) across all four training densities (5, 10, 15, 20). Performance consistently improves as larger proportions of outliers are removed, with gains saturating around the 8%–10% range. At 10% removal, SHARP-QoS achieves an average improvement of 81.84% compared to the setting where outliers are not addressed.

Table 6.7: Impact of Cold-start on WSDREAM-1 on RT and TP dataset.

Training Density	QoS Parameter	Type ↓ CSP →	MAE				RMSE			
			0	5	10	20	0	5	10	20
10	RT	CB	0.3243	0.3908	0.4304	0.5007	1.2450	1.4103	1.5700	1.6888
		CS		0.3582	0.3783	0.4371		1.3330	1.4521	1.5392
		CU		0.3666	0.3835	0.4000		1.3432	1.4223	1.4402
	TP	CB	11.4814	14.7406	18.6156	27.7954	41.7156	51.1179	56.5571	77.0098
		CS		15.8290	16.2723	19.0722		53.2600	54.8492	62.8405
		CU		12.7160	12.9015	14.7178		46.1673	46.9849	51.3352
20	RT	CB	0.2930	0.3588	0.4073	0.4588	1.1516	1.3151	1.4440	1.5427
		CS		0.3288	0.3515	0.4067		1.2559	1.3253	1.4634
		CU		0.3399	0.3482	0.3658		1.2768	1.2966	1.3368
	TP	CB	10.4069	13.6560	17.0083	19.9448	38.7746	49.6500	54.9375	62.6531
		CS		13.4201	14.4766	21.9069		48.8123	49.5523	64.9157
		CU		11.9470	12.6056	13.1524		43.0584	45.5650	46.2888

6.3.2.3 Cold-start Sensitivity Analysis

Cold-start represents new users or services that lack historical QoS records. QoS prediction methods that rely mainly on QoS-specific features often struggle in such situations. SHARP-QoS incorporates public contextual attributes, including AS and region information, to deal with this.

To assess the sensitivity of our model to cold-start cases, we create three scenarios by randomly selecting a fixed percentage (CSP) of users and services and removing all their corresponding invocation entries from the QoS matrices: *(i) Cold-start User (CU):* A fixed CSP of users is selected, and all their QoS entries are removed, *(ii) Cold-start Service (CS):* A fixed CSP of services is selected, and all their QoS entries are removed, and*(iii) Cold-start Both (CB):* A fixed CSP of both users and services is selected, and all their corresponding entries are removed. All other data remain unchanged.

Table 6.7 presents the model performance under two training densities (10, 20) for WSDREAM-1 (RT, TP) dataset. The main observations are as follows:

(i) As CSP increases (0-20, with a step-size of 5), performance declines for both RT and TP due to the reduction in available training data.

(ii) For a fixed CSP, higher training density leads to better performance.

(iii) Under the same CSP, the CB scenario shows the largest performance drop because

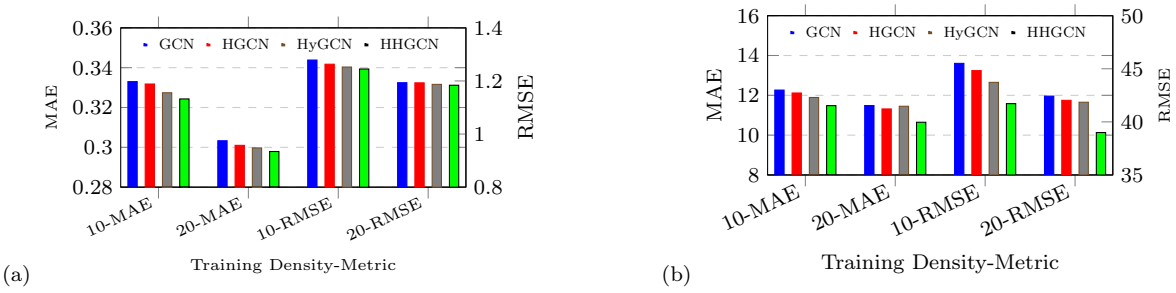


Figure 6.3: Impact of HHGCN module on WSDREAM-1 (a) RT, (b) TP dataset.

more entries are removed compared to CU and CS.

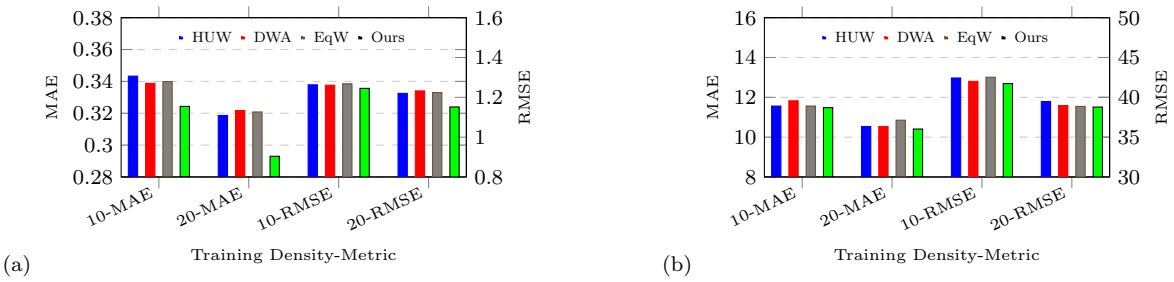
These results show that SHARP-QoS demonstrates reasonable robustness in handling cold-start cases.

6.3.2.4 Model Ablation Study

This section analyzes the contribution of two key components of SHARP-QoS: *(i)* the hierarchical feature extraction via HHGCN, and *(ii)* the EMA-based loss scaling.

6.3.2.4.1 Impact of Hierarchical Feature Extraction Block Fig. 6.3 demonstrates that the proposed hyperbolic–hypergraph convolution (HHGCN) consistently outperforms the other three variants: standard Euclidean graph convolution (GCN), Euclidean hypergraph convolution (HGCN), and hyperbolic graph convolution (HyGCN). On the WSDREAM-1 (RT) dataset, HHGCN achieves improvements of 3.00% (MAE) and 3.07% (RMSE) over GCN, and 2.46% (MAE) and 2.44% (RMSE) over HGCN. Moreover, incorporating user/service hyper-edges yields additional gains over HyGCN (1.59% MAE and 1.79% RMSE). A more pronounced improvement is observed on the TP dataset: HHGCN surpasses GCN by 7.89% (MAE) and 8.47% (RMSE), HGCN by 6.64% (MAE) and 8.47% (RMSE), and HyGCN by 6.28% (MAE) and 5.96% (RMSE).

These results demonstrate that HHGCN effectively captures hierarchical structure for QoS data and exploits hyper-edge relations to strengthen representation learning. Note that, HyGCN ablations for AS and RG graphs are omitted, as HyGCN was directly adopted based on empirical validation.



(a) (b)

Figure 6.4: Impact of EMA-based loss balancing on WSDREAM-1 (a) RT, and (b) TP dataset.

Table 6.8: Performance across single- and multi-task learning on WSDREAM-1.

Framework	Task	MAE				RMSE			
		5	10	15	20	5	10	15	20
Single-Task	RT	0.4490	0.3684	0.3478	0.3311	1.4366	1.3343	1.3199	1.2843
SHARP-QoS		0.3668	0.3243	0.3099	0.2930	1.3116	1.2450	1.1979	1.1516
Relative Gain (%)		18.31	11.96	10.90	11.51	8.70	6.69	9.24	10.33
Single-Task	TP	14.0499	11.8974	11.7706	11.0872	50.3999	42.6373	41.9440	39.5894
SHARP-QoS		13.2402	11.4814	10.8035	10.4069	47.0426	41.7156	39.7392	38.7746
Relative Gain (%)		5.76	3.50	8.22	6.14	6.66	2.16	5.26	2.06

6.3.2.4.2 Impact of EMA-Based Loss Scaling Fig. 6.4 compares different loss-balancing strategies, including homoscedastic uncertainty weighting (HUW) [123], dynamic weight averaging (DWA) [125], equal weighting (EqW), and proposed EMA-based approach. On WSDREAM-1 (RT, TP), our method outperforms HUW by 6.78% (MAE) and 3.50% (RMSE), and DWA by 6.60% (MAE) and 3.91% (RMSE). Moreover, our approach further offers an additional 6.62% MAE and 3.86% RMSE improvement compared to EqW on both QoS parameters.

This shows that the EMA-based strategy adapts quickly to task fluctuations, stabilizes task weighting, thereby mitigating negative transfer, and ensures balanced optimization, making it a more reliable loss-balancing strategy for joint QoS prediction.

6.3.2.4.3 Single-task vs Multi-task Performance To analyze the impact of learning multiple QoS parameters simultaneously and assess the effect of negative transfer in SHARP-QoS, we perform an additional experiment comparing single-task

Table 6.9: Module Ablation on WSDREAM-1 RT and TP dataset.

Module	RT				TP			
	MAE		RMSE		MAE		RMSE	
	10	20	10	20	10	20	10	20
- HHGCNs - Cross-SNR	0.6229	0.6049	2.1301	1.9588	31.5699	31.2864	96.6124	96.1364
- SNR - Cross-SNR	0.5275	0.5139	1.8966	1.8783	28.8214	25.6720	83.9696	76.4003
- HHGCNs - SNR	0.3309	0.3258	1.2664	1.1987	12.7072	18.8945	47.6640	62.3913
- SNR	0.3442	0.3154	1.2777	1.2237	15.0648	14.0172	52.2302	48.9402
- HHGCNs	0.3395	0.3187	1.2728	1.2221	17.5674	16.6736	58.8570	57.0896
- Cross-SNR	0.3345	0.3031	1.2638	1.1987	13.1844	12.0310	48.9919	43.9704
Ours	**0.3243**	**0.2930**	**1.2450**	**1.1516**	**11.4814**	**10.4069**	**41.7156**	**38.7746**

and multi-task learning performance. As shown in Table 6.8, SHARP-QoS consistently outperforms its single-task counterpart across both QoS parameters (RT and TP) under multiple training densities in terms of both MAE and RMSE. The observed relative improvements indicate that the proposed multi-task framework, benefits from positive cross-task knowledge sharing rather than suffering from negative transfer.

Collectively, these analyses demonstrate that SHARP-QoS maintains stable and effective multi-task learning behavior while mitigating harmful task interference across multiple QoS objectives.

6.3.2.5 Module Ablation Study

Table 6.9 presents the module-wise ablation on the WSDREAM-1 dataset (RT, TP). A clear performance decline is observed across all ablated variants, reinforcing the necessity of each component. The full framework, comprising shared representations and structural features, consistently delivers the best performance. Removing the Cross-SNR module results in a noticeable drop in average accuracy (2.70% on RT, 12.34% on TP), indicating that different QoS parameters exhibit informative patterns that benefit one another. Eliminating either the HHGCNs or SNR blocks also degrades performance, showing that these components supply complementary signals, QoS features via HHGCNs capture fine-grained service behavior, while contextual cues using SNR alleviate the sparsity and cold-start problem. The highest degradation occurs when two modules are removed (HHGCNs+SNR, SNR+Cross-SNR or HHGCNs+Cross-SNR), where removing HHGCNs+SNR yields smaller average drops

Table 6.10: Impact of different graphs on WSDREAM-1 RT and TP dataset

Model	RT				TP			
	MAE		RMSE		MAE		RMSE	
	10	20	10	20	10	20	10	20
QoS	0.3318	0.3108	1.2598	1.2105	12.7465	11.7712	48.0036	42.3711
QoS + AS	0.3299	0.3035	1.2605	1.1898	12.1578	11.7534	45.5000	42.9646
QoS + RG	0.3290	0.3000	1.2592	1.1885	12.0947	11.4153	43.0615	41.9777
Single Graph	0.3262	0.3003	1.2516	1.1852	11.8777	11.3378	43.3086	41.3478
Ours	0.3243	0.2930	1.2450	1.1516	11.4814	10.4069	41.7156	38.7746

(4.17% RT and 25.27% TP).

These results highlight that all three modules contribute substantially to performance, and their joint integration achieves the most accurate joint QoS prediction.

6.3.2.6 Impact of Context Graphs

Table 6.10 evaluates the contribution of complementary contextual signals derived from AS and RG graphs. When HHGCNs are applied only to the QoS invocation graphs, the average performance drops by 3.51% in RT and 10.78% in TP relative to the full model performance, indicating that only QoS-based structural features are insufficient alone to achieve the best performance. Introducing either the AS or RG context graph improves performance, with the RG graph providing a stronger gain. This is likely due to the RG graph containing approximately 4.37× more edges, thereby offering richer collaborative cues. Furthermore, collapsing all edges into a single merged graph (a unified adjacency matrix) results in inferior performance compared to our design, which processes the QoS, AS, and RG graphs separately using HHGCN and HyGCN modules. The merged graph combines heterogeneous structural signals into a single message-passing process, potentially amplifying noise and weakening fine-grained feature propagation.

In contrast, treating the context graphs independently exploits their complementary roles: the RG graph captures coarse geographical proximity, while the AS graph encodes fine-grained routing and peering relationships. This isolated yet complementary modeling yields the optimal performance.

Table 6.11: Confidence Intervals on WSDREAM-1.

Conf. level	RT-10	RT-20	TP-10	TP-20
90%	(0.2856, 0.3629)	(0.2699, 0.3293)	(10.9175, 11.9258)	(9.9840, 10.9102)
95%	(0.2782, 0.3703)	(0.2643, 0.3350)	(10.8210, 12.0224)	(9.8954, 10.9988)
99%	(0.2638, 0.3847)	(0.2531, 0.3461)	(10.6322, 12.2112)	(9.7219, 11.1722)
MAE±Std	0.3243±0.1773	0.2996±0.1362	11.4217±2.3139	10.4471±2.1253

Conf.: Confidence, Std: Standard deviation

6.3.2.7 Statistical Significance Analysis

To ensure the reliability of SHARP-QoS, we perform a statistical significance analysis [171]. Specifically, we partition the test prediction errors into $G = 50$ equally sized, non-overlapping groups and compute the MAE of each group independently. Using these group-wise errors, we estimate the empirical mean $\bar{m}$ and standard deviation s. We then compute the two-sided confidence intervals (CI) using Eq. 6.29 for confidence levels $\alpha \in \{90\%, 95\%, 99\%\}$ with their corresponding z-scores z_α, and present the results in Table 6.11.

$$\mathrm{CI}_\alpha = \left[\bar{m} - z_\alpha \frac{s}{\sqrt{G}},\ \bar{m} + z_\alpha \frac{s}{\sqrt{G}}\right] \tag{6.29}$$

Hypothesis Validation: We assess stability by verifying whether the observed MAE falls within the confidence bounds $H_0 :\ \mathrm{MAE}_{\mathrm{obs}} \in \mathrm{CI}_\alpha$. If satisfied, H_0 is accepted, indicating reliable performance; otherwise, if rejected, suggesting potential instability.

As shown in Table 6.11, the observed MAE values (last row) for both RT and TP consistently fall within the computed confidence intervals across all confidence levels. Moreover, the confidence intervals narrow with increasing data density, indicating lower variance and improved reliability, thereby demonstrating that the prediction behavior across subgroups is statistically sound.

6.4 Summary

This chapter presents a unified framework for joint QoS prediction called SHARP-QoS that leverages hierarchical features via hyperbolic convolution networks, adaptive multi-context routing using QoS and contextual features, and an EMA-based

loss balancing strategy. This enables higher-order, complex features while allowing for flexible feature sharing across QoS parameters, and effectively mitigating negative transfer and enhancing graph representation learning. Experimental results show that our method achieves superior prediction accuracy and inference latency compared to existing approaches. Future work will explore real-time service environments, deployment on resource-constrained devices, and integration of online measurement signals for dynamic graph updates.

6.5 Limitation

SHARP-QoS presents a joint multi-QoS prediction framework that improves robustness by mitigating negative transfer via EMA-based loss scale balancing. It further enhances representation learning by exploiting hierarchical structure dependencies inherent in QoS interactions and contextual attributes (e.g., geographical regions and network topology). Despite these advancements, several challenges remain:

(i) SHARP-QoS is primarily designed for static QoS modeling and does not explicitly incorporate temporal dynamics. As QoS behavior evolves over time due to changing network conditions and service workloads, extending SHARP-QoS to a time-aware framework remains an important direction for improving prediction accuracy in dynamic environments.

(ii) Although SHARP-QoS improves hierarchical representation learning, it does not explicitly model trustworthiness or credibility of users and services. In particular, grey-sheep users or services with atypical invocation patterns are not handled separately, which may affect prediction reliability under anomalous conditions.

Addressing these limitations by integrating temporal modeling, and anomaly-resilient hierarchical representations forms an important direction for future research toward developing more reliable and adaptive multi-QoS prediction frameworks.

Chapter 7

Conclusion & Future Directions

7.1 Summary of Contributions

This thesis addressed the problem of reliable Quality of Service (QoS) prediction in dynamic service-oriented computing environments, where accurate estimation of service performance is essential for dependable recommendation, service selection, orchestration, and autonomous decision-making. Real-world QoS prediction remains fundamentally challenging due to extreme sparsity, temporal variability, anomalous observations, grey-sheep behavior, cold-start conditions, and the growing need to jointly model multiple interdependent QoS attributes. Although prior studies have explored these issues individually, existing solutions often treat them as isolated problems, resulting in fragmented frameworks with limited robustness and scalability. Motivated by these limitations, this thesis investigated how reliable QoS prediction can be progressively improved through integrated learning frameworks that jointly consider representation learning, structural dependencies, temporal evolution, anomaly resilience, and multi-task optimization.

A key insight emerging from this thesis is that QoS prediction cannot be effectively addressed through isolated modeling assumptions. Instead, reliable prediction requires progressively richer representations of user–service interactions together with mechanisms that explicitly account for temporal dynamics, credibility of observations, and interdependencies among multiple QoS objectives. The four frameworks developed in

this thesis collectively reflect this conceptual progression.

The initial framework, TPMCF, originated from the observation that conventional collaborative filtering and matrix factorization approaches are fundamentally limited by their static nature. Real-world QoS behavior evolves continuously over time due to changing workloads, network conditions, and user mobility. This led to the insight that QoS prediction is inherently a spatio-temporal learning problem rather than a purely relational estimation problem. By integrating graph convolutional matrix factorization with transformer-based temporal modeling, TPMCF demonstrated the importance of jointly capturing collaborative structural dependencies and temporal evolution patterns. More importantly, this framework established that expressive representation learning forms the foundation for reliable QoS estimation in dynamic service ecosystems.

However, improved representation learning alone proved insufficient for dependable prediction in realistic environments. During the development of ARRQP, it became evident that real-world QoS observations are frequently corrupted by noisy measurements, unstable service behavior, outliers, and irregular invocation patterns associated with grey-sheep users or services. These irregularities destabilize collaborative learning and significantly degrade predictive consistency. This led to a second major insight of the thesis: robustness and credibility-awareness are not secondary enhancements but central requirements for practical QoS prediction systems. ARRQP therefore extended the earlier spatio-temporal learning perspective toward anomaly-resilient prediction through multi-head graph convolution modeling, residual learning, and credibility-aware optimization. The framework demonstrated that predictive accuracy and robustness must be addressed jointly, particularly in sparse and noisy service environments.

As the research progressed further, another limitation became apparent. Although graph-based approaches improve relational learning, conventional pairwise interaction modeling remains insufficient for representing the complex collaborative dependencies present in service ecosystems. User behavior, service characteristics, and temporal contexts interact simultaneously in ways that cannot be fully captured through sim-

ple graph structures. This realization motivated the development of HCTN and introduced a third major insight of the thesis: reliable QoS prediction requires higher-order relational modeling. By integrating hypergraph learning with dual-channel transformer architectures, HCTN moved beyond pairwise connectivity toward unified modeling of triadic and higher-order interactions among users, services, and temporal contexts. More significantly, HCTN unified representation learning, temporal dynamics, sparsity mitigation, and credibility-aware learning within a single coherent architecture. This framework demonstrated that the various challenges of QoS prediction are deeply interconnected rather than independent optimization objectives.

The final stage of the thesis further expanded the scope of QoS prediction from single-attribute estimation toward holistic multi-QoS learning. Earlier frameworks primarily focused on predicting individual QoS parameters such as response time or throughput independently. However, practical service-oriented environments rarely evaluate services using a single performance criterion. Real-world service selection depends simultaneously on multiple interdependent QoS attributes, often exhibiting heterogeneous scales and conflicting optimization objectives. This motivated the development of SHARP-QoS and led to another important insight: multi-QoS prediction is fundamentally different from isolated single-task estimation because it requires balancing shared collaborative knowledge with task-specific discriminative learning. Through sparsely-gated hierarchical adaptive routing, hierarchical feature sharing, and EMA-based loss balancing, SHARP-QoS demonstrated how multi-task architectures can effectively capture inter-QoS relationships while mitigating negative transfer. This progression transformed the overall perspective of the thesis from isolated QoS estimation toward integrated service intelligence modeling.

Taken together, the progression across the four frameworks reveals several broader principles regarding dependable QoS prediction. First, expressive representation learning is essential for overcoming sparsity and improving generalization in large-scale service environments. Second, anomaly resilience and credibility-aware learning are inseparable from predictive reliability in practical deployments. Third, higher-order collaborative dependencies provide substantially richer information than conventional

pairwise interaction modeling. Fourth, temporal dynamics are intrinsic characteristics of QoS behavior and must be incorporated directly into the learning process. Finally, joint QoS prediction requires carefully balanced architectures capable of simultaneously leveraging shared knowledge and preserving task-specific behavior.

Beyond methodological advancements, the thesis also reflects the evolving role of QoS prediction within modern intelligent infrastructures. As service-oriented ecosystems increasingly support edge computing, Internet of Things environments, autonomous systems, and distributed AI applications, reliable QoS estimation becomes closely tied to system responsiveness, operational stability, and dependable autonomous decision-making. In such environments, QoS prediction is no longer merely a recommendation problem but an enabling component for adaptive and trustworthy service orchestration. The frameworks developed in this thesis contribute toward this broader objective by providing scalable, robust, and temporally-aware learning architectures suitable for dynamic real-world deployment scenarios.

In summary, this thesis advances the state-of-the-art in QoS prediction through a coherent progression of frameworks that collectively address representation learning, sparsity mitigation, anomaly resilience, higher-order relational modeling, temporal dynamics, and joint multi-QoS optimization. Rather than viewing these challenges independently, the thesis demonstrates that reliable QoS prediction emerges from their unified treatment within integrated learning architectures. The proposed frameworks therefore, establish a principled foundation for next-generation QoS prediction systems capable of supporting dependable, adaptive, and intelligent service ecosystems in increasingly dynamic and large-scale computing environments.

7.2 Future Research Directions

While this thesis advances QoS prediction through progressively more robust, expressive, and unified learning frameworks, several important research directions remain open for further investigation. The proposed frameworks, namely TPMCF, ARRQP, HCTN, and SHARP-QoS, collectively address temporal dynamics, anomaly resilience,

higher-order representation learning, and joint multi-QoS optimization. However, the increasing complexity of modern distributed service ecosystems introduces additional challenges related to scalability, deployment realism, orchestration intelligence, continual adaptation, fairness, and privacy preservation.

7.2.1 Unified Reliability-Aware Spatio-Temporal Multi-QoS Learning

Although the proposed frameworks progressively address temporal modeling, anomaly resilience, higher-order interaction learning, and multi-QoS prediction, these capabilities are still developed through sequential architectural evolution. A promising future direction is the development of fully unified end-to-end learning frameworks capable of jointly modeling temporal evolution, structural dependencies, reliability estimation, anomaly resilience, and multi-QoS optimization within a single coherent architecture.

Such frameworks may integrate dynamic credibility estimation, adaptive anomaly-aware feature weighting, hypergraph representation learning, and hyperbolic geometric embeddings to better capture the complex interactions among users, services, contexts, and temporal conditions. Developing such unified learning systems would further improve robustness, scalability, and generalization in highly dynamic service-oriented environments.

7.2.2 Continual Learning in Dynamic QoS-aware Ecosystems

Real-world service ecosystems continuously evolve due to changing workloads, service updates, network fluctuations, user mobility, and emerging application demands. Most existing QoS prediction approaches, including the frameworks proposed in this thesis, primarily operate under offline or periodically retrained learning settings. Consequently, model adaptation under continuously evolving distributions remains an important open challenge.

Future research may investigate continual and online QoS prediction frameworks capable of incrementally adapting to new users, services, and interaction patterns with-

out catastrophic forgetting. Meta-learning, streaming graph neural networks, adaptive transformers, and memory-augmented learning strategies may provide promising directions for enabling efficient and stable lifelong QoS learning in large-scale dynamic environments.

7.2.3 Large-Scale Real-World Deployment and Dataset Development

The experimental evaluation conducted in this thesis demonstrates the effectiveness of the proposed frameworks across benchmark datasets and real-world QoS scenarios. However, widely used benchmark datasets such as WSDREAM were collected under earlier service-oriented architectures and may not fully capture the complexity of modern cloud-native, edge-native, and microservice-based computing environments.

Future work will therefore focus on developing and releasing large-scale QoS datasets derived from contemporary distributed systems, including gRPC-based microservice ecosystems, edge-cloud infrastructures, intelligent transportation systems, and autonomous service platforms. Such datasets are expected to capture richer contextual dependencies, dynamic service orchestration behavior, fine-grained temporal variability, distributed resource contention, and realistic anomaly patterns observed in modern service deployments.

Extensive validation on such contemporary large-scale environments will further strengthen the generalizability and practical applicability of the proposed QoS prediction frameworks.

7.2.4 QoS-aware Service Selection and Autonomous Service Orchestration

The QoS prediction frameworks proposed in this thesis primarily focus on accurate estimation of service performance characteristics. However, QoS prediction represents only one foundational component within broader service management and orchestration pipelines. Practical deployment of intelligent service ecosystems additionally

requires mechanisms for service selection, adaptive composition, resource allocation, runtime monitoring, and decision optimization.

Future research may therefore extend the proposed frameworks toward fully integrated QoS-aware orchestration systems that combine predictive modeling with reinforcement learning, multi-objective optimization, scheduling policies, and adaptive resource management strategies. Such systems could dynamically select and compose services based on predicted QoS behavior while continuously adapting to changing network conditions and workload variations.

These extensions would be particularly valuable in latency-sensitive and safety-critical applications such as autonomous vehicles, smart transportation systems, edge intelligence platforms, and distributed AI services, where reliable and adaptive service orchestration is essential for operational efficiency and system stability.

7.2.5 Fairness-Aware, Bias-Resilient, and Privacy-Preserving QoS Prediction

While the proposed frameworks explicitly address sparsity, grey-sheep behavior, and anomalous QoS observations to improve predictive robustness, broader concerns related to fairness, bias mitigation, and privacy preservation remain important future research directions. In large-scale collaborative environments, minority interaction patterns, cold-start users, or sparsely represented services may be systematically underrepresented during model optimization, potentially introducing unintended prediction bias.

Future work may investigate fairness-aware QoS learning frameworks capable of ensuring equitable predictive performance across heterogeneous user and service groups while preserving robustness and scalability. In parallel, privacy-preserving QoS prediction methods based on federated learning, distributed optimization, differential privacy, and secure collaborative learning may enable reliable QoS modeling without directly exposing sensitive user interaction data.

Incorporating explainable learning mechanisms and interpretable reliability esti-

mation into QoS prediction frameworks also represents an important direction for improving transparency, trustworthiness, and deployment readiness in safety-critical and autonomous service ecosystems.

7.3 Final Remarks

This thesis advances the field of QoS prediction through the development of a coherent sequence of learning frameworks that progressively address the major challenges of modern service-oriented computing environments, including temporal dynamics, sparsity, anomaly resilience, higher-order interaction modeling, and joint multi-QoS learning. Rather than treating these challenges independently, the thesis demonstrates that reliable QoS prediction emerges from their unified treatment within expressive and robust learning architectures.

The progression from TPMCF to ARRQP, HCTN, and SHARP-QoS reflects an evolving understanding of QoS prediction as a fundamentally structured, temporal, credibility-aware, and multi-objective learning problem. The proposed frameworks collectively establish the importance of integrating representation learning, anomaly resilience, higher-order collaborative modeling, and adaptive multi-task optimization for dependable QoS estimation in dynamic large-scale service ecosystems.

Beyond improving predictive accuracy, this thesis highlights the broader role of QoS prediction within intelligent distributed systems. As cloud-native infrastructures, edge computing platforms, autonomous systems, and AI-driven service ecosystems continue to evolve, reliable QoS prediction increasingly becomes a critical enabling component for adaptive orchestration, trustworthy automation, and dependable service management.

The methodologies and insights developed in this thesis therefore provide not only robust solutions for QoS prediction, but also a foundational basis for future research in intelligent service orchestration, adaptive distributed computing, trustworthy AI-driven infrastructure management, and large-scale autonomous service ecosystems.

Bibliography

[1] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS Evaluation for Real-World Web Services," in *IEEE ICWS 2010, Miami, Florida, USA, July 5-10, 2010*. IEEE Computer Society, 2010, pp. 83–90. [Online]. Available: https://doi.org/10.1109/ICWS.2010.10

[2] "Web Services Description Language (WSDL)," in *Encyclopedia of Social Network Analysis and Mining*, 2014, p. 2405. [Online]. Available: https://doi.org/10.1007/978-1-4614-6170-8_100914

[3] T. Erl, *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR, 2004.

[4] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions," in *4th International Conference on Web Information Systems Engineering, WISE 2003, Rome, Italy, December 10-12, 2003*. IEEE Computer Society, 2003, pp. 3–12. [Online]. Available: https://doi.org/10.1109/WISE.2003.1254461

[5] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004. [Online]. Available: https://doi.org/10.1109/TSE.2004.11

[6] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *UAI '98: Proceedings of the Fourteenth Conference on UAI, University of Wisconsin Business*

School, Madison, Wisconsin, USA, July 24-26, 1998, G. F. Cooper and S. Moral, Eds. Morgan Kaufmann, 1998, pp. 43–52. [Online]. Available: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=231&proceeding_id=14

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 285–295. [Online]. Available: https://doi.org/10.1145/371920.372071

[8] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140–152, 2011. [Online]. Available: https://doi.org/10.1109/TSC.2010.52

[9] G. White, A. Palade, and S. Clarke, "QoS Prediction for Reliable Service Composition in IoT," in *Service-Oriented Computing - ICSOC 2017 Workshops - ASOCA, ISyCC, WESOACS, and Satellite Events, Málaga, Spain, November 13-16, 2017, Revised Selected Papers*, ser. Lecture Notes in Computer Science, L. Braubach, J. M. Murillo, N. Kaviani, M. Lama, L. Burgueño, N. Moha, and M. Oriol, Eds., vol. 10797. Springer, 2017, pp. 149–160. [Online]. Available: https://doi.org/10.1007/978-3-319-91764-1_12

[10] Y. Shen, J. Zhu, X. Wang, L. Cai, X. Yang, and B. Zhou, "Geographic Location-Based Network-Aware QoS Prediction for Service Composition," in *2013 IEEE 20th ICWS, Santa Clara, CA, USA, June 28 - July 3, 2013*. IEEE Computer Society, 2013, pp. 66–74. [Online]. Available: https://doi.org/10.1109/ICWS.2013.19

[11] Z. Zheng, X. Li, M. Tang, F. Xie, and M. R. Lyu, "Web Service QoS Prediction via Collaborative Filtering: A Survey," *IEEE Trans.*

Serv. Comput., vol. 15, no. 4, pp. 2455–2472, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2020.2995571

[12] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service QoS prediction," *Knowledge-Based Systems*, vol. 138, pp. 188–201, 2017.

[13] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-Aware Deep Collaborative Filtering for Service Recommendation," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 6, pp. 3796–3807, 2021. [Online]. Available: https://doi.org/10.1109/TSMC.2019.2931723

[14] J. Zhou, D. Ding, Z. Wu, and Y. Xiu, "Spatial Context-Aware Time-Series Forecasting for QoS Prediction," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 2, pp. 918–931, 2023. [Online]. Available: https://doi.org/10.1109/TNSM.2023.3250512

[15] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware qos prediction approach for web service recommendation," *Knowl. Based Syst.*, vol. 115, pp. 55–65, 2017. [Online]. Available: https://doi.org/10.1016/j.knosys.2016.09.033

[16] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "QoS Prediction of Web Services Based on Two-Phase K-Means Clustering," in *2015 IEEE ICWS 2015, New York, NY, USA, June 27 - July 2, 2015*, J. A. Miller and H. Zhu, Eds. IEEE Computer Society, 2015, pp. 161–168. [Online]. Available: https://doi.org/10.1109/ICWS.2015.31

[17] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services," in *IEEE 22nd ISSRE 2011, Hiroshima, Japan, November 29 - December 2, 2011*, T. Dohi and B. Cukic, Eds. IEEE Computer Society, 2011, pp. 210–219. [Online]. Available: https://doi.org/10.1109/ISSRE.2011.17

[18] G. Zou, T. Li, M. Jiang, S. Hu, C. Cao, B. Zhang, Y. Gan, and Y. Chen, "DeepTSQP: Temporal-aware service QoS prediction via deep neural network and feature integration," *Knowledge-Based Systems*, vol. 241, p. 108062, 2022.

[19] P. Zhang, Y. Chen, W. Huang, H. Zhu, and Q. Zhao, "Generative-Adversarial-Based Feature Compensation to Predict Quality of Service," *IEEE Trans. Serv. Comput.*, vol. 17, no. 1, pp. 209–223, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2023.3335296

[20] C. Tang, S. Zhao, B. Chen, X. Lu, and Q. Zhang, "A two-dimensional time-aware cloud service recommendation approach with enhanced similarity and trust," *J. Parallel Distributed Comput.*, vol. 190, p. 104889, 2024. [Online]. Available: https://doi.org/10.1016/j.jpdc.2024.104889

[21] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-Aware QoS Prediction With Neural Collaborative Filtering for Internet-of-Things Services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2956827

[22] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reputation-Aware QoS Value Prediction of Web Services," in *2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*. IEEE Computer Society, 2013, pp. 41–48. [Online]. Available: https://doi.org/10.1109/SCC.2013.43

[23] S. H. Ghafouri, S. M. Hashemi, and P. C. K. Hung, "A Survey on Web Service QoS Prediction Methods," *IEEE Trans. Serv. Comput.*, vol. 15, no. 4, pp. 2439–2454, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2020.2980793

[24] G. Zou, M. Jiang, S. Niu, H. Wu, S. Pang, and Y. Gan, "QoS-Aware Web Service Recommendation with Reinforced Collaborative Filtering," in *Service-Oriented Computing - 16th ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings*, ser. Lecture Notes in Computer Science, C. Pahl,

M. Vukovic, J. Yin, and Q. Yu, Eds., vol. 11236. Springer, 2018, pp. 430–445. [Online]. Available: https://doi.org/10.1007/978-3-030-03596-9_31

[25] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowl. Based Syst.*, vol. 23, no. 6, pp. 520–528, 2010. [Online]. Available: https://doi.org/10.1016/j.knosys.2010.03.009

[26] R. R. Chowdhury, S. Chattopadhyay, and C. Adak, "CAHPHF: Context-Aware Hierarchical QoS Prediction With Hybrid Filtering," *IEEE Trans. Serv. Comput.*, vol. 15, no. 4, pp. 2232–2247, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2020.3041626

[27] Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng, "An Embedding Based Factorization Machine Approach for Web Service QoS Prediction," in *Service-Oriented Computing - 15th ICSOC 2017, Malaga, Spain, November 13-16, 2017, Proceedings*, ser. Lecture Notes in Computer Science, E. M. Maximilien, A. Vallecillo, J. Wang, and M. Oriol, Eds., vol. 10601. Springer, 2017, pp. 272–286. [Online]. Available: https://doi.org/10.1007/978-3-319-69035-3_19

[28] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering," *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 573–579, 2013. [Online]. Available: https://doi.org/10.1109/TSC.2012.31

[29] X. Wu, B. Cheng, and J. Chen, "Collaborative Filtering Service Recommendation Based on a Novel Similarity Computation Method," *IEEE Trans. Serv. Comput.*, vol. 10, no. 3, pp. 352–365, 2017. [Online]. Available: https://doi.org/10.1109/TSC.2015.2479228

[30] Z. Chen, L. Shen, F. Li, D. You, and J. P. B. Mapetu, "Web service QoS prediction: when collaborative filtering meets data fluctuating in big-range," *World Wide Web*, vol. 23, no. 3, pp. 1715–1740, 2020. [Online]. Available: https://doi.org/10.1007/s11280-020-00787-x

[31] Y. Shi, K. Zhang, B. Liu, and L. Cui, "A New QoS Prediction Approach Based on User Clustering and Regression Algorithms," in *IEEE ICWS 2011, Washington, DC, USA, July 4-9, 2011*. IEEE Computer Society, 2011, pp. 726–727. [Online]. Available: https://doi.org/10.1109/ICWS.2011.95

[32] D. D. Lee and H. S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[33] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*, Y. Lechevallier and G. Saporta, Eds. Physica-Verlag, 2010, pp. 177–186. [Online]. Available: https://doi.org/10.1007/978-3-7908-2604-3_16

[34] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank SVD via fast alternating least squares," *J. Mach. Learn. Res.*, vol. 16, pp. 3367–3402, 2015. [Online]. Available: https://dl.acm.org/doi/10.5555/2789272.2912106

[35] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," in *NIPS*, 2007, p. 1257–1264.

[36] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An Extended Matrix Factorization Approach for QoS Prediction in Service Selection," in *2012 IEEE Ninth International Conference on Services Computing, Honolulu, HI, USA, June 24-29, 2012*, L. E. Moser, M. Parashar, and P. C. K. Hung, Eds. IEEE Computer Society, 2012, pp. 162–169. [Online]. Available: https://doi.org/10.1109/SCC.2012.36

[37] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, 2013. [Online]. Available: https://doi.org/10.1109/TSC.2011.59

[38] K. Qi, H. Hu, W. Song, J. Ge, and J. Lu, "Personalized QoS Prediction via Matrix Factorization Integrated with Neighborhood Information," in *2015 IEEE International Conference on Services Computing, SCC 2015, New York City, NY, USA, June 27 - July 2, 2015*. IEEE Computer Society, 2015, pp. 186–193. [Online]. Available: https://doi.org/10.1109/SCC.2015.34

[39] H. Wu, K. Yue, B. Li, B. Zhang, and C. Hsu, "Collaborative QoS prediction with context-sensitive matrix factorization," *Future Gener. Comput. Syst.*, vol. 82, pp. 669–678, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.06.020

[40] Y. Xu, J. Yin, and W. Lo, "A Unified Framework of QoS-Based Web Service Recommendation with Neighborhood-Extended Matrix Factorization," in *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications, Koloa, HI, USA, December 16-18, 2013*. IEEE Computer Society, 2013, pp. 198–205. [Online]. Available: https://doi.org/10.1109/SOCA.2013.10

[41] Y. Yin, S. Aihua, G. Min, Y. Xu, and W. Shuoping, "QoS Prediction for Web Service Recommendation with Network Location-Aware Neighbor Selection," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, no. 4, pp. 611–632, 2016. [Online]. Available: https://doi.org/10.1142/S0218194016400040

[42] F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang, "Outlier-Resilient Web Service QoS Prediction," in *Proceedings of the Web Conference 2021*, ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3099–3110. [Online]. Available: https://doi.org/10.1145/3442381.3449938

[43] Y. Zhang, C. Yin, Z. Lu, D. Yan, M. Qiu, and Q. Tang, "Recurrent tensor factorization for time-aware service recommendation," *Appl. Soft Comput.*, vol. 85, 2019. [Online]. Available: https://doi.org/10.1016/j.asoc.2019.105762

[44] M. Li, Q. Lu, and M. Zhang, "A Two-Tier Service Filtering Model for Web Service QoS Prediction," *IEEE Access*, vol. 8, pp. 221 278–221 287, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3043773

[45] Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng, "An Embedding Based Factorization Machine Approach for Web Service QoS Prediction," in *Service-Oriented Computing - 15th ICSOC 2017, Malaga, Spain, November 13-16, 2017, Proceedings*, ser. Lecture Notes in Computer Science, E. M. Maximilien, A. Vallecillo, J. Wang, and M. Oriol, Eds., vol. 10601. Springer, 2017, pp. 272–286. [Online]. Available: https://doi.org/10.1007/978-3-319-69035-3_19

[46] L. Shen, M. Pan, L. Liu, D. You, F. Li, and Z. Chen, "Contexts Enhance Accuracy: On Modeling Context Aware Deep Factorization Machine for Web API QoS Prediction," *IEEE Access*, vol. 8, pp. 165 551–165 569, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3022891

[47] G. Zou, J. Chen, Q. He, K. Li, B. Zhang, and Y. Gan, "NDMF: Neighborhood-Integrated Deep Matrix Factorization for Service QoS Prediction," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2717–2730, 2020. [Online]. Available: https://doi.org/10.1109/TNSM.2020.3027185

[48] Y. Yin, H. Xu, T. Liang, M. Chen, H. Gao, and A. Longo, "Leveraging Data Augmentation for Service QoS Prediction in Cyber-physical Systems," *ACM Trans. Internet Techn.*, vol. 21, no. 2, pp. 35:1–35:25, 2021. [Online]. Available: https://doi.org/10.1145/3425795

[49] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, and Z. Mai, "QoS Prediction for Service Recommendation With Features Learning in Mobile Edge Computing Environment," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1136–1145, 2020. [Online]. Available: https://doi.org/10.1109/TCCN.2020.3027681

[50] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng, and Y. Yang, "Covering-Based Web Service Quality Prediction via Neighborhood-Aware Matrix Factorization," *IEEE Trans. Serv. Comput.*, vol. 14, no. 5, pp. 1333–1344, 2021. [Online]. Available: https://doi.org/10.1109/TSC.2019.2891517

[51] P. Zhang, W. Huang, Y. Chen, M. Zhou, and Y. Al-Turki, "A Novel Deep-Learning-Based QoS Prediction Model for Service Recommendation

Utilizing Multi-Stage Multi-Scale Feature Fusion With Individual Evaluations," *IEEE Trans Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1740–1753, 2024. [Online]. Available: https://doi.org/10.1109/TASE.2023.3244184

[52] Z. Jia, L. Jin, Y. Zhang, C. Liu, K. Li, and Y. Yang, "Location-Aware Web Service QoS Prediction via Deep Collaborative Filtering," *IEEE Trans. Comput. Soc. Syst.*, vol. 10, no. 6, pp. 3524–3535, 2023. [Online]. Available: https://doi.org/10.1109/TCSS.2022.3217277

[53] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[54] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[55] M. Liu, H. Xu, Q. Z. Sheng, and Z. Wang, "QoSGNN: Boosting QoS Prediction Performance With Graph Neural Networks," *IEEE Trans. Serv. Comput.*, vol. 17, no. 2, pp. 645–658, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2023.3343351

[56] Z. Chang, D. Ding, and Y. Xia, "A graph-based QoS prediction approach for web service recommendation," *Appl. Intell.*, vol. 51, no. 10, pp. 6728–6742, 2021. [Online]. Available: https://doi.org/10.1007/s10489-020-02120-5

[57] H. Liu, Z. Zhang, H. Li, Q. Wu, and Y. Zhang, "Large Language Model Aided QoS Prediction for Service Recommendation," in *IEEE ICSSE, SSE*

2025, Helsinki, Finland, July 7-12, 2025, R. N. Chang, C. K. Chang, J. Yang, N. Atukorala, D. Chen, S. Helal, S. Tarkoma, Q. He, T. Kosar, C. A. Ardagna, J. Berrocal, K. E. Maghaouri, and Y. Sun, Eds. IEEE, 2025, pp. 116–127. [Online]. Available: https://doi.org/10.1109/SSE67621.2025.00023

[58] S. Chattopadhyay, R. Chanda, S. Kumar, and C. Adak, "Offdq: An offline deep learning framework for qos prediction," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1987–1996. [Online]. Available: https://doi.org/10.1145/3485447.3512107

[59] G. Zou, S. Wu, S. Hu, C. Cao, Y. Gan, B. Zhang, and Y. Chen, "NCRL: neighborhood-based collaborative residual learning for adaptive qos prediction," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 2030–2043, 2023. [Online]. Available: https://doi.org/10.1109/TSC.2022.3213129

[60] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model," *Decis. Support Syst.*, vol. 107, no. C, p. 103–115, Mar. 2018. [Online]. Available: https://doi.org/10.1016/j.dss.2017.12.012

[61] G. Zou, Y. Huang, S. Hu, Y. Gan, B. Zhang, and Y. Chen, "TRCF: Temporal Reinforced Collaborative Filtering for Time-Aware QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1847–1860, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2023.3329110

[62] E. Tong, W. Niu, and J. Liu, "A Missing QoS Prediction Approach via Time-Aware Collaborative Filtering," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3115–3128, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2021.3103769

[63] E. Jawabreh and A. Taweel, "Enhanced Time-Aware Collaborative Filtering for QoS Web Service Prediction," in *Service-Oriented and Cloud Computing - 10th*

IFIP WG 6.12 European Conference, ESOCC 2023, Larnaca, Cyprus, October 24-25, 2023, Proceedings, ser. Lecture Notes in Computer Science, G. A. Papadopoulos, F. Rademacher, and J. Soldani, Eds., vol. 14183. Springer, 2023, pp. 70–83. [Online]. Available: https://doi.org/10.1007/978-3-031-46235-1_5

[64] X. Zhu, X. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-Maintaining Privacy Preservation and Location-Aware Low-Rank Matrix Factorization for QoS Prediction Based Web Service Recommendation," *IEEE Trans. Serv. Comput.*, vol. 14, no. 3, pp. 889–902, 2021. [Online]. Available: https://doi.org/10.1109/TSC.2018.2839741

[65] F. Chen, Y. Du, W. Zhong, and H. Wang, "Web Service QoS Prediction Based on Reputation and Location Aware Matrix Factorization," in *IEEE SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta 2022, Haikou, China, December 15-18, 2022*. IEEE, 2022, pp. 1722–1729. [Online]. Available: https://doi.org/10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00245

[66] D. Ryu, K. Lee, and J. Baik, "Location-Based Web Service QoS Prediction via Preference Propagation to Address Cold Start Problem," *IEEE Trans. Serv. Comput.*, vol. 14, no. 3, pp. 736–746, 2021. [Online]. Available: https://doi.org/10.1109/TSC.2018.2821686

[67] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[68] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *CoRR*, vol. abs/1710.10903, 2017. [Online]. Available: http://arxiv.org/abs/1710.10903

[69] Z. Wu, D. Ding, Y. Xiu, Y. Zhao, and J. Hong, "Robust qos prediction based on reputation integrated graph convolution network," *IEEE Trans. Serv. Comput.*, vol. 17, no. 3, pp. 1154–1167, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2023.3317642

[70] H. Wu, S. Tian, B. Jin, Y. Zhao, and L. Zhang, "Effective Graph Modeling and Contrastive Learning for Time-Aware QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3513–3526, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2024.3478836

[71] J. Zhu, B. Li, J. Wang, D. Li, Y. Liu, and Z. Zhang, "BGCL: Bi-subgraph network based on graph contrastive learning for cold-start QoS prediction," *Knowl. Based Syst.*, vol. 263, p. 110296, 2023. [Online]. Available: https://doi.org/10.1016/j.knosys.2023.110296

[72] S. Hu, G. Zou, B. Zhang, S. Wu, S. Lin, Y. Gan, and Y. Chen, "Temporal-Aware QoS Prediction via Dynamic Graph Neural Collaborative Learning," in *Service-Oriented Computing - 20th ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings*, ser. Lecture Notes in Computer Science, J. Troya, B. Medjahed, M. Piattini, L. Yao, P. Fernández, and A. Ruiz-Cortés, Eds., vol. 13740. Springer, 2022, pp. 125–133. [Online]. Available: https://doi.org/10.1007/978-3-031-20984-0_8

[73] K. Sharma, Y. Lee, S. Nambi, A. Salian, S. Shah, S. Kim, and S. Kumar, "A Survey of Graph Neural Networks for Social Recommender Systems," *ACM Comput. Surv.*, vol. 56, no. 10, p. 265, 2024. [Online]. Available: https://doi.org/10.1145/3661821

[74] M. I. Smahi, F. Hadjila, C. Tibermacine, and A. Benamar, "A deep learning approach for collaborative prediction of Web service QoS," *Serv. Oriented Comput. Appl.*, vol. 15, no. 1, pp. 5–20, 2021. [Online]. Available: https://doi.org/10.1007/s11761-020-00304-y

[75] C. Yu and L. Huang, "CluCF: a clustering CF algorithm to address data sparsity problem," *Serv. Oriented Comput. Appl.*, vol. 11, no. 1, pp. 33–45, 2017. [Online]. Available: https://doi.org/10.1007/s11761-016-0191-8

[76] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization," *IEEE Trans. Parallel Distributed Syst.*, vol. 28, no. 10, pp. 2911–2924, 2017. [Online]. Available: https://doi.org/10.1109/TPDS.2017.2700796

[77] S. Li, J. Wen, F. Luo, T. Cheng, and Q. Xiong, "A Location and Reputation Aware Matrix Factorization Approach for Personalized Quality of Service Prediction," in *2017 IEEE ICWS 2017, Honolulu, HI, USA, June 25-30, 2017*, I. Altintas and S. Chen, Eds. IEEE, 2017, pp. 652–659. [Online]. Available: https://doi.org/10.1109/ICWS.2017.78

[78] X. Chen, Y. Du, Y. Han, J. Huang, and Z. Qian, "Hybrid Reputation Fusion and Mutual Information Maximization for Web Services QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. 18, no. 6, pp. 3878–3891, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3623459

[79] B. Li, C. Ye, X. Yu, H. Zhou, and C. Huang, "Qos prediction based on temporal information and request context," *Serv. Oriented Comput. Appl.*, vol. 15, no. 3, pp. 231–244, 2021. [Online]. Available: https://doi.org/10.1007/s11761-021-00322-4

[80] M. Wang, B. Liu, J. Li, Y. Li, H. Chen, Z. Zhou, and W. Zhang, "A Location-Based Approach for Web Service QoS Prediction via Multivariate Time Series Forecast," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020, pp. 36–39.

[81] Z. Wang, Y. Xiao, C. Sun, W. Zheng, and X. Jiao, "Location-Aware Feature Interaction Learning for Web Service Recommendation," in *2020 IEEE ICWS 2020, Beijing, China, October 19-23, 2020*. IEEE, 2020, pp. 232–239. [Online]. Available: https://doi.org/10.1109/ICWS49710.2020.00037

[82] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative Web Service QoS Prediction with Location-Based Regularization," in *2012 IEEE 19th ICWS Honolulu, HI, USA, June 24-29, 2012*, C. A. Goble, P. P. Chen, and J. Zhang, Eds. IEEE Computer Society, 2012, pp. 464–471. [Online]. Available: https://doi.org/10.1109/ICWS.2012.49

[83] M. Wu, Q. Lu, and Y. Wang, "A Multi-stack Denoising Autoencoder for QoS Prediction," in *Artificial Neural Networks and Machine Learning - 31st ICANN 2022, Bristol, UK, September 6-9, 2022, Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, and M. Aydin, Eds., vol. 13530. Springer, 2022, pp. 757–768. [Online]. Available: https://doi.org/10.1007/978-3-031-15931-2_62

[84] M. Chen, J. Yu, J. Wang, and X. Li, "HTG: A heterogeneous topology aware model to improve cold start in cloud service QoS prediction," *Trans. Emerg. Telecommun. Technol.*, vol. 35, no. 3, Mar. 2024. [Online]. Available: https://doi.org/10.1002/ett.4951

[85] Y. Yin, Q. Di, J. Wan, and T. Liang, "Time-Aware Smart City Services Based on QoS Prediction: A Contrastive Learning Approach," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 18 745–18 753, 2023. [Online]. Available: https://doi.org/10.1109/JIOT.2023.3281869

[86] B. Gras, A. Brun, and A. Boyer, "Identifying Grey Sheep Users in Collaborative Filtering: A Distribution-Based Technique," in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, ser. UMAP '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 17–26. [Online]. Available: https://doi.org/10.1145/2930238.2930242

[87] Z. Wang, X. Zhang, M. Yan, L. Xu, and D. Yang, "HSA-Net: Hidden-State-Aware Networks for High-Precision QoS Prediction," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 6, pp. 1421–1435, 2022. [Online]. Available: https://doi.org/10.1109/TPDS.2021.3111810

[88] P. Zhang, J. Fan, Y. Chen, W. Huang, H. Zhu, and Q. Zhao, "An end-to-end deep learning qos prediction model based on temporal context and feature fusion," *IEEE Trans. Serv. Comput.*, vol. 18, no. 3, pp. 1232–1244, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3562324

[89] H. Zhang, M. Wu, Q. Feng, and H. Li, "AERQP: adaptive embedding representation-based qos prediction for web service recommendation," *J. Supercomput.*, vol. 80, no. 3, pp. 3042–3065, 2024. [Online]. Available: https://doi.org/10.1007/s11227-023-05582-9

[90] G. Zou, S. Lin, S. Hu, S. Duan, Y. Gan, B. Zhang, and Y. Chen, "FHC-DQP: Federated Hierarchical Clustering for Distributed QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4073–4086, 2023. [Online]. Available: https://doi.org/10.1109/TSC.2023.3309257

[91] G. Zou, W. Yu, S. Hu, Y. Gan, B. Zhang, and Y. Chen, "FRLN: Federated Residual Ladder Network for Data-Protected QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. 17, no. 3, pp. 963–976, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2024.3377100

[92] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. Hsu, "Multiple Attributes QoS Prediction via Deep Neural Model with Contexts," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 1084–1096, 2021. [Online]. Available: https://doi.org/10.1109/TSC.2018.2859986

[93] Z. Wang, X. Zhang, Z. S. Li, and M. Yan, "QoSBERT: An Uncertainty-Aware Approach Based on Pretrained Language Models for Service Quality Prediction," *IEEE Trans. Serv. Comput.*, vol. 18, no. 6, pp. 4096–4108, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3620138

[94] Y. Xiu, D. Ding, Z. Wu, Y. Zhao, and J. Liu, "Dual heterogeneous graph contrastive learning for qos prediction," *Appl. Intell.*, vol. 55, no. 7, p. 566, 2025. [Online]. Available: https://doi.org/10.1007/s10489-025-06431-3

[95] Z. Wang, X. Zhang, Z. S. Li, S. Huang, and M. Yan, "Feature noise resilient for qos prediction with probabilistic deep supervision," *IEEE Trans. Serv. Comput.*, vol. 18, no. 4, pp. 2062–2074, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3577425

[96] Q. Wang, C. Yan, J. Wang, W. Zheng, and Y. Xiao, "Location-aware Collaborative Filtering and Feature Interaction Learning for QoS Prediction," in *IEEE ICWS 2024, Shenzhen, China, July 7-13, 2024*. IEEE, 2024, pp. 291–299. [Online]. Available: https://doi.org/10.1109/ICWS62655.2024.00051

[97] H. Liu, Z. Zhang, H. Li, Q. Wu, and Y. Zhang, "Large language model aided qos prediction for service recommendation," in *IEEE ICSSE, SSE 2025, Helsinki, Finland, July 7-12, 2025*, R. N. Chang, C. K. Chang, J. Yang, N. Atukorala, D. Chen, S. Helal, S. Tarkoma, Q. He, T. Kosar, C. A. Ardagna, J. Berrocal, K. E. Maghaouri, and Y. Sun, Eds. IEEE, 2025, pp. 116–127. [Online]. Available: https://doi.org/10.1109/SSE67621.2025.00023

[98] J. Mi and H. Wu, "RNL: A Robust and Highly-Efficient Model for Time-Aware Web Service QoS Prediction," in *Advanced Intelligent Computing Technology and Applications: 19th International Conference, ICIC 2023, Zhengzhou, China, August 10–13, 2023, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 27–39. [Online]. Available: https://doi.org/10.1007/978-981-99-4752-2_3

[99] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal Pattern-Aware QoS Prediction via Biased Non-Negative Latent Factorization of Tensors," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1798–1809, 2020. [Online]. Available: https://doi.org/10.1109/TCYB.2019.2903736

[100] P. Tang, T. Ruan, H. Wu, and X. Luo, "Temporal pattern-aware QoS prediction by Biased Non-negative Tucker Factorization of tensors," *Neurocomputing*, vol. 582, p. 127447, 2024. [Online]. Available: https://doi.org/10.1016/j.neucom.2024.127447

[101] X. Luo, M. Chen, H. Wu, Z. Liu, H. Yuan, and M. Zhou, "Adjusting Learning Depth in Nonnegative Latent Factorization of Tensors for Accurately Modeling Temporal Patterns in Dynamic QoS Data," *IEEE Trans Autom. Sci. Eng.*, vol. 18, no. 4, pp. 2142–2155, 2021. [Online]. Available: https://doi.org/10.1109/TASE.2020.3040400

[102] S. Wang, Y. Ma, B. Cheng, F. Yang, and R. N. Chang, "Multi-Dimensional QoS Prediction for Service Recommendations," *IEEE Trans. Serv. Comput.*, vol. 12, no. 1, pp. 47–57, 2019. [Online]. Available: https://doi.org/10.1109/TSC.2016.2584058

[103] A. Chai, M. Li, H. Yang, and C. Guo, "Emd-emlstm: A qos analysis and prediction method for industrial internet of things," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 32 730–32 744, 2024. [Online]. Available: https://doi.org/10.1109/JIOT.2024.3375855

[104] Y. Xia, L. Wang, and H. Wu, "A Non-Negative Snowflake Factorization of Tensors Model for Dynamic QoS Prediction," in *2025 Joint International Conference on Automation-Intelligence-Safety (ICAIS) & International Symposium on Autonomous Systems (ISAS)*, 2025, pp. 1–6.

[105] W. Li, M. Lin, X. Xu, L. Lin, Z. Xu, and X. Luo, "Neural nonnegative latent factorization of tensors model with acceleration and unconstraint," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 56, no. 1, pp. 164–178, 2026. [Online]. Available: https://doi.org/10.1109/TSMC.2025.3622727

[106] H. Wu and X. Luo, "Instance-Frequency-Weighted Regularized, Nonnegative and Adaptive Latent Factorization of Tensors for Dynamic QoS Analysis," in *2021 IEEE ICWS 2021, Chicago, IL, USA, September 5-10, 2021*, C. K. Chang, E. Daminai, J. Fan, P. Ghodous, M. Maximilien, Z. Wang, R. Ward, and J. Zhang, Eds. IEEE, 2021, pp. 560–568. [Online]. Available: https://doi.org/10.1109/ICWS53863.2021.00077

[107] H. Wu, X. Luo, and M. Zhou, "Advancing non-negative latent factorization of tensors with diversified regularization schemes," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1334–1344, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2020.2988760

[108] S. Kumar and S. Chattopadhyay, "TRQP: Trust-Aware Real-Time QoS Prediction Framework Using Graph-Based Learning," in *Service-Oriented Computing - 20th International Conference, ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings*, ser. Lecture Notes in Computer Science, J. Troya, B. Medjahed, M. Piattini, L. Yao, P. Fernández, and A. Ruiz-Cortés, Eds., vol. 13740. Springer, 2022, pp. 143–152. [Online]. Available: https://doi.org/10.1007/978-3-031-20984-0_10

[109] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation Forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy.* IEEE Computer Society, 2008, pp. 413–422. [Online]. Available: https://doi.org/10.1109/ICDM.2008.17

[110] X. Chen, Y. Du, G. Tang, F. Chen, Y. Luo, and H. Wang, "A QoS Prediction Framework via Utility Maximization and Region-Aware Matrix Factorization," *IEEE Trans. Serv. Comput.*, vol. 18, no. 2, pp. 557–571, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3541554

[111] T. Lu, X. Zhang, Z. Wang, and M. Yan, "A feature distribution smoothing network based on gaussian distribution for qos prediction," in *IEEE ICWS 2023, Chicago, IL, USA, July 2-8, 2023*, C. A. Ardagna, B. Benatallah, H. Bian, C. K. Chang, R. N. Chang, J. Fan, G. C. Fox, Z. Jin, X. Liu, H. Ludwig, M. Sheng, and J. Yang, Eds. IEEE, 2023, pp. 687–694. [Online]. Available: https://doi.org/10.1109/ICWS60048.2023.00087

[112] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization," *IEEE*

Trans. Reliab., vol. 65, no. 1, pp. 28–37, 2016. [Online]. Available: https://doi.org/10.1109/TR.2015.2464075

[113] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A Data-Characteristic-Aware Latent Factor Model for Web Services QoS Prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2525–2538, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2020.3014302

[114] H. S. M. Muslim, S. Rubab, M. M. Khan, N. Iltaf, A. K. Bashir, and K. Javed, "S-RAP: relevance-aware QoS prediction in web-services and user contexts," *Knowl. Inf. Syst.*, vol. 64, no. 7, pp. 1997–2022, 2022. [Online]. Available: https://doi.org/10.1007/s10115-022-01699-0

[115] S. Li, J. Wen, F. Luo, and G. Ranzi, "Time-Aware QoS Prediction for Cloud Service Recommendation Based on Matrix Factorization," *IEEE Access*, vol. 6, pp. 77 716–77 724, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2883939

[116] W. Zhang, H. Sun, X. Liu, and X. Guo, "Temporal QoS-aware web service recommendation via non-negative tensor factorization," in *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, C. Chung, A. Z. Broder, K. Shim, and T. Suel, Eds. ACM, 2014, pp. 585–596. [Online]. Available: https://doi.org/10.1145/2566486.2568001

[117] L. Ding *et al.*, "Joint QoS Prediction for Web Services based on Deep Fusion of Features," *IEEE TNSM*, pp. 1–1, 2023.

[118] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. K. Hung, "Personalized LSTM Based Matrix Factorization for Online QoS Prediction," in *2018 IEEE ICWS 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE, 2018, pp. 34–41. [Online]. Available: https://doi.org/10.1109/ICWS.2018.00012

[119] M. Wang, Z. Wang, and S. Tang, "MF-Informer for long-term QoS prediction in edge-cloud collaboration environments," *Wirel. Networks*, vol. 30, no. 8, pp.

7497–7512, 2024. [Online]. Available: https://doi.org/10.1007/s11276-024-03670-z

[120] R. Caruana, "Multitask Learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734

[121] H. Luo, "Gated multi-task learning for qos-aware service recommendation," in *Journal of Physics: Conference Series*, vol. 2547, no. 1. IOP Publishing, 2023, p. 012011.

[122] M. Chen *et al.*, "HTG: A heterogeneous topology aware model to improve cold start in cloud service QoS prediction," *Trans. Emerg. Telecommun. Technol.*, vol. 35, no. 3, Mar. 2024.

[123] H. Lian, J. Li, H. Wu, Y. Zhao, L. Zhang, and X. Wang, "Toward Effective Personalized Service QoS Prediction From the Perspective of Multi-Task Learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 3, pp. 2587–2597, 2023. [Online]. Available: https://doi.org/10.1109/TNSM.2023.3236348

[124] W. Liang, Y. Li, J. Xu, Z. Qin, D. Zhang, and K. Li, "QoS Prediction and Adversarial Attack Protection for Distributed Services Under DLaaS," *IEEE Trans. Computers*, vol. 73, no. 3, pp. 669–682, 2024. [Online]. Available: https://doi.org/10.1109/TC.2021.3077738

[125] J. Wang, X. Zhang, Q. Wang, W. Zheng, and Y. Xiao, "QoS Prediction Method via Multi-Task Learning for Web Service Recommendation," in *IEEE ICWS 2024, Shenzhen, China, July 7-13, 2024*. IEEE, 2024, pp. 1353–1355. [Online]. Available: https://doi.org/10.1109/ICWS62655.2024.00167

[126] S. Wang, C. Hsu, Z. Liang, Q. Sun, and F. Yang, "Multi-user web service selection based on multi-qos prediction," *Inf. Syst. Frontiers*, vol. 16, no. 1, pp. 143–152, 2014. [Online]. Available: https://doi.org/10.1007/s10796-013-9455-4

[127] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 32–39, 2014. [Online]. Available: https://doi.org/10.1109/TSC.2012.34

[128] P. Zhang, J. Ren, W. Huang, Y. Chen, Q. Zhao, and H. Zhu, "A Deep-Learning Model for Service QoS Prediction Based on Feature Mapping and Inference," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1311–1325, 2024. [Online]. Available: https://doi.org/10.1109/TSC.2023.3326208

[129] C. Wu, W. Qiu, X. Wang, Z. Zheng, and X. Yang, "Time-Aware and Sparsity-Tolerant QoS Prediction Based on Collaborative Filtering," in *IEEE ICWS 2016, San Francisco, CA, USA, June 27 - July 2, 2016*, S. Reiff-Marganiec, Ed. IEEE Computer Society, 2016, pp. 637–640. [Online]. Available: https://doi.org/10.1109/ICWS.2016.88

[130] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[131] G. White, A. Palade, C. Cabrera, and S. Clarke, "Autoencoders for QoS Prediction at the Edge," in *2019 IEEE International Conference on Pervasive Computing and Communications, PerCom, Kyoto, Japan, March 11-15, 2019*. IEEE, 2019, pp. 1–9. [Online]. Available: https://doi.org/10.1109/PERCOM.2019.8767397

[132] H. V. Nguyen and L. Bai, "Cosine Similarity Metric Learning for Face Verification," in *Computer Vision - ACCV 2010 - 10th Asian Conference on Computer Vision, Queenstown, New Zealand, November 8-12, 2010, Revised*

Selected Papers, Part II, ser. Lecture Notes in Computer Science, R. Kimmel, R. Klette, and A. Sugimoto, Eds., vol. 6493. Springer, 2010, pp. 709–720. [Online]. Available: https://doi.org/10.1007/978-3-642-19309-5_55

[133] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010. [Online]. Available: https://dl.acm.org/doi/10.5555/1756006.1953039

[134] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on AIML*, vol. 14, no. 3, pp. 1–159, 2020.

[135] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2627435.2670313

[136] X. Li, Q. Lu, Y. Dong, and D. Tao, "Robust Subspace Clustering by Cauchy Loss Function," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2067–2078, 2019.

[137] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[138] X. Chen, Z. Han, Y. Wang, Q. Zhao, D. Meng, and Y. Tang, "Robust Tensor Factorization with Unknown Noise," in *2016 IEEE Conference on CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 5213–5221. [Online]. Available: https://doi.org/10.1109/CVPR.2016.563

[139] T. Liang, M. Chen, Y. Yin, L. Zhou, and H. Ying, "Recurrent Neural Network Based Collaborative Filtering for QoS Prediction in IoV," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2400–2410, 2022. [Online]. Available: https://doi.org/10.1109/TITS.2021.3099346

[140] J. Zhu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "CARP: Context-Aware Reliability Prediction of Black-Box Web Services," in *2017 IEEE ICWS 2017, Honolulu, HI, USA, June 25-30, 2017*, I. Altintas and S. Chen, Eds. IEEE, 2017, pp. 17–24. [Online]. Available: https://doi.org/10.1109/ICWS.2017.10

[141] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, 2009. [Online]. Available: https://doi.org/10.1109/MSP.2008.930649

[142] P. J. Huber, *Robust Estimation of a Location Parameter*. Springer New York, 1992, pp. 492–518.

[143] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[144] F. Bi, T. He, Y. Xie, and X. Luo, "Two-Stream Graph Convolutional Network-Incorporated Latent Feature Analysis," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 3027–3042, 2023. [Online]. Available: https://doi.org/10.1109/TSC.2023.3241659

[145] Z. Zhao, Z. Yang, C. Li, Q. Zeng, W. Guan, and M. Zhou, "Dual Feature Interaction-Based Graph Convolutional Network," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9019–9030, 2023. [Online]. Available: https://doi.org/10.1109/TKDE.2022.3220789

[146] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica, "Pylot: A Modular Platform for Exploring Latency-Accuracy Tradeoffs in Autonomous Vehicles," in *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 8806–8813. [Online]. Available: https://doi.org/10.1109/ICRA48506.2021.9561747

[147] H. Gao *et al.*, "Context-Aware QoS Prediction With Neural Collaborative Filtering for Internet-of-Things Services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, 2020.

[148] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS Prediction for Service Recommendation with Deep Feature Learning in Edge Computing Environment," *Mob. Networks Appl.*, vol. 25, no. 2, pp. 391–401, 2020. [Online]. Available: https://doi.org/10.1007/s11036-019-01241-7

[149] Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng, "An embedding based factorization machine approach for web service QoS prediction," in *ICSOC*, 2017, pp. 272–286.

[150] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A Data-Characteristic-Aware Latent Factor Model for Web Services QoS Prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2525–2538, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2020.3014302

[151] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[152] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative Web Service Quality Prediction via Exploiting Matrix Factorization and Network Map," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 1, pp. 126–137, 2016. [Online]. Available: https://doi.org/10.1109/TNSM.2016.2517097

[153] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A Data-Aware Latent Factor Model for Web Service QoS Prediction," in *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part I*, ser. Lecture

Notes in Computer Science, Q. Yang, Z. Zhou, Z. Gong, M. Zhang, and S. Huang, Eds., vol. 11439. Springer, 2019, pp. 384–399. [Online]. Available: https://doi.org/10.1007/978-3-030-16148-4_30

[154] S. Chawla and A. Gionis, "k-means-: A Unified Approach to Clustering and Outlier Detection," in *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*. SIAM, 2013, pp. 189–197. [Online]. Available: https://doi.org/10.1137/1.9781611972832.21

[155] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, W. Chen, J. F. Naughton, and P. A. Bernstein, Eds. ACM, 2000, pp. 93–104. [Online]. Available: https://doi.org/10.1145/342009.335388

[156] K. Chen, H. Mao, X. Shi, Y. Xu, and A. Liu, "Trust-Aware and Location-Based Collaborative Filtering for Web Service QoS Prediction," in *41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, Turin, Italy, July 4-8, 2017. Volume 2*. IEEE Computer Society, 2017, pp. 143–148. [Online]. Available: https://doi.org/10.1109/COMPSAC.2017.8

[157] M. Lin, H. C. L. Jr., and G. Shmueli, "Research Commentary - Too Big to Fail: Large Samples and the p-Value Problem," *Inf. Syst. Res.*, vol. 24, no. 4, pp. 906–917, 2013. [Online]. Available: https://doi.org/10.1287/isre.2013.0480

[158] A. Hameed, J. Violos, A. Leivadeas, N. Santi, R. Grünblatt, and N. Mitton, "Toward QoS Prediction Based on Temporal Transformers for IoT Applications," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 4, pp. 4010–4027, 2022. [Online]. Available: https://doi.org/10.1109/TNSM.2022.3217170

[159] S. Kumar, S. Chattopadhyay, and C. Adak, "TPMCF: Temporal QoS Prediction Using Multi-Source Collaborative Features," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 4, pp. 3945–3955, 2024. [Online]. Available: https://doi.org/10.1109/TNSM.2024.3395428

[160] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021. [Online]. Available: https://doi.org/10.1109/TNNLS.2020.2978386

[161] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," in *Proceedings of the 32nd ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 2048–2057.

[162] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. J. Bota, H. Liu, T. Schultz, and H. Gamboa, "TSFEL: Time Series Feature Extraction Library," *SoftwareX*, vol. 11, p. 100456, 2020. [Online]. Available: https://doi.org/10.1016/j.softx.2020.100456

[163] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.

[164] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proceedings of the Twenty-Seventh IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, J. Lang, Ed. ijcai.org, 2018, pp. 3634–3640. [Online]. Available: https://doi.org/10.24963/ijcai.2018/505

[165] J. Bi, Z. Wang, H. Yuan, X. Wu, R. Wu, J. Zhang, and M. Zhou, "Long-Term Water Quality Prediction With Transformer-Based Spatial-Temporal Graph Fusion," *IEEE Trans Autom. Sci. Eng.*, vol. 22, pp. 11 392–11 404, 2025. [Online]. Available: https://doi.org/10.1109/TASE.2025.3535415

[166] J. Ma, Z. Zhao, J. Chen, A. Li, L. Hong, and E. H. Chi, "SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-Task Learning," in *The Thirty-Third AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 216–223.

[167] S. Kumar and S. Chattopadhyay, "ARRQP: Anomaly Resilient Real-Time QoS Prediction Framework With Graph Convolution," *IEEE Trans. Serv. Comput.*, vol. 18, no. 3, pp. 1245–1261, 2025. [Online]. Available: https://doi.org/10.1109/TSC.2025.3565376

[168] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic Neural Networks," in *NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 5350–5360.

[169] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic Graph Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 4869–4880.

[170] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through L_0 Regularization," *CoRR*, vol. abs/1712.01312, 2017.

[171] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.