ST-DGL: Spatio-Temporal Dynamic Graph Learning for Mobile Edge QoS Prediction

Guobing Zou*, Fei Zhao*, Shengxiang Hu*, Song Yang*, Shengye Pang*,✉, Yanglan Gan†,✉, Bofeng Zhang‡
*School of Computer Engineering and Science, Shanghai University, Shanghai, China
{gbzou, feizhao, shengxianghu, yangsong, pangsy}@shu.edu.cn
†School of Computer Science and Technology, Donghua University, Shanghai, China
ylgan@dhu.edu.cn
‡School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai, China
bfzhang@sspu.edu.cn

***Abstract*—Predicting Quality of Service (QoS) in mobile edge environments remains challenging because most existing research relies on static models or simple time-series prediction. However, in mobile edge environments, users are constantly moving, and the resources of edge servers vary accordingly, exhibiting strong spatio-temporal dependencies. Traditional approaches struggle to accurately capture these dynamics. Moreover, widely used datasets such as WS-DREAM do not model realistic user mobility, base station distribution, or the temporal characteristics of edge resource availability, which further limits their applicability to real-world mobile edge environments. To address these challenges, this study utilizes the public CHESTNUT dataset [1], which features realistic user mobility and server workload dynamics. Based on this dataset, we propose a novel framework named Spatio-Temporal Dynamic Graph Learning (ST-DGL), which jointly models edge server spatiotemporal features, user mobility, and service attributes to capture complex dependencies in mobile edge environments. Experimental results on the dataset demonstrate that ST-DGL achieves the best performance under medium-to-high QoS densities (10%–20%) and remains competitive under extremely sparse settings (5%). These results provide new research insights and practical support for dynamic QoS prediction in mobile edge computing environments.**

***Index Terms*—Mobile Edge Computing; QoS Prediction; Spatio-Temporal Graph Learning; User Mobility; Edge Server Feature Extraction**

I. Introduction

With the rapid development and widespread adoption of the Internet of Things (IoT), an increasing number of applications impose stringent requirements on multiple QoS metrics, such as ultra-low latency, high reliability, low jitter, and service availability. These applications include autonomous driving, industrial automation, immersive virtual reality (VR), augmented reality (AR), telemedicine, and intelligent transportation systems [2]. In these critical scenarios, QoS performance directly determines system safety and stability—industrial automation and remote surgery demand millisecond-level latency, while autonomous driving is highly sensitive to latency jitter and service disruptions. Traditional centralized cloud architectures face fundamental challenges in meeting these stringent QoS demands due to long transmission distances and potential backhaul congestion, introducing significant delays and reliability issues. To address this, Mobile Edge Computing (MEC) has emerged, deploying computing, storage, and networking resources in close proximity to end-users and IoT devices. This architectural shift significantly reduces physical transmission distances and bypasses backbone bottlenecks, markedly improving key QoS metrics like end-to-end latency, responsiveness, and service reliability for mission-critical applications.

However, mobile edge computing's distributed and low-latency nature demands accurate QoS prediction to ensure the stability and user experience of real-time applications [3]. In mobile edge environments, user mobility, time-varying wireless links, and changing traffic patterns create highly dynamic network conditions. Since edge servers have limited resources and small coverage areas, user movement frequently triggers request switching across servers [4], causing rapid workload changes and pronounced QoS fluctuations. Without accurate advance prediction of these variations, effective resource scheduling, task offloading, and service migration become difficult, degrading performance and the continuity of latency-sensitive services [5].

Traditional QoS prediction methods have evolved from memory-based collaborative filtering [6] and matrix factorization [7] to deep learning [8] and graph neural networks [9]. Although time-series features have been incorporated to uncover latent correlations [10], these approaches typically focus on filling temporal gaps in centralized matrices rather than predicting future values, and fundamentally fail to generalize to highly dynamic mobile edge environments where rapid user mobility and network fluctuations severely degrade predictive accuracy.

Consequently, research has shifted toward edge-specific solutions, integrating user-edge server similarity [11] and context-aware relevance [12]. Notable recent advancements include Pred_{QoS} [13], which captures temporal dynamics via VQ-VAE, and MEC-RDESN [14], which leverages echo state networks to adapt to real-time mobility and handoffs. However, despite these improvements in capturing user-centric and temporal dynamics, existing edge prediction models still exhibit deficiencies in modeling the realistic dynamics of the

edge infrastructure itself. First, they fail to explicitly incorporate edge-specific contextual factors, particularly the dynamic fluctuations in server workloads. These variations in resource contention and load conditions have a profound impact on QoS performance [15], yet current approaches lack mechanisms to capture these dynamics. Second, the effects of different times of the day, such as rush hours or off-peak periods, are not adequately considered. Changes in user activity and service demand during these times lead to fluctuating system load, resulting in unpredictable QoS variations.

To address the limitations outlined above, we propose a novel framework for edge QoS prediction called Spatio-Temporal Dynamic Graph Learning (ST-DGL). This model is specifically designed to capture the non-linear spatio-temporal dependencies among edge server workloads, primarily driven by user mobility and the resultant fluctuations in service flow. In particular, it leverages mobility-aware server contextual features from the past k timestamps to construct dynamic spatio-temporal graphs. The model then performs spatio-temporal feature extraction and fusion on two critical temporal features of servers—the load rate (L) and the throughput (T)—capturing essential workload dynamics for more accurate QoS prediction. We validate the framework's effectiveness in dynamic edge environments using CHESTNUT, a specialized mobile edge QoS dataset that provides the necessary complexity for evaluating spatio-temporal models and has been adopted in recent representative studies to ensure a rigorous comparative performance analysis.

The main contributions of this paper are:

- We propose a novel framework ST-DGL for mobile edge QoS prediction. It learns mobility-driven spatio-temporal correlations among edge servers by modeling their dynamic interactions, thereby improving the accuracy of next-timestamp QoS prediction in mobile edge environments.
- We propose a dynamic edge server spatio-temporal feature extraction method. Specifically, by leveraging user mobility and server location relationships over the recent k timestamps, we generate dynamic localized spatio-temporal graphs. Based on these graphs, we perform an L-layer spatio-temporal extraction and fusion on critical server workload features, namely load rate and throughput. Integrating these enriched dynamic representations with static attributes effectively enhances QoS prediction accuracy in dynamic mobile edge environments.
- We conducted extensive experiments using the dataset to evaluate the proposed ST-DGL framework. The results show that ST-DGL outperforms competing baselines in QoS prediction accuracy, demonstrating superior performance in both dense and sparse edge scenarios.

The remainder of this paper is organized as follows. Section II defines the problem and preliminaries. Section III elaborates on our ST-DGL framework. Section IV presents the dataset size and experimental results. Section V reviews related work, Section VI concludes the paper.

II. Problem Formulation

Definition 1 (Mobile Edge Service Ecosystem): A mobile edge service ecosystem is defined as a five-tuple $\mathcal{M} = \langle \mathcal{U}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{U} = \{u_1, u_2, ...\}$ represents a set of n users, $\mathcal{E} = \{e_1, e_2, ...\}$ denotes a set of m edge servers, $\mathcal{S} = \{s_1, s_2, ...\}$ denotes a set of services deployed on edge servers (i.e., edge services), and $\mathcal{T} = \{t_1, t_2, \ldots\}$ is a set of consecutive timestamps. The set $\mathcal{R} = \{r^t_{u,e,s} \mid u \in \mathcal{U}, e \in \mathcal{E}, s \in \mathcal{S}, t \in \mathcal{T}\}$ consists of user–server–service QoS invocations across multiple timestamps.

User mobility and edge server characteristics significantly impact QoS prediction accuracy.

Definition 2 (User Mobility): In a mobile edge service ecosystem $\mathcal{M}$, users are characterized by spatio-temporal mobility features that vary over time. Let $\mathcal{X} = \{\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^{|\mathcal{T}|}\}$ denote the collection of user feature sets across all timestamps. At timestamp $t \in \mathcal{T}$, the user feature set is defined as $\mathcal{X}^t = \{x^t_1, x^t_2, \ldots, x^t_{|\mathcal{U}|}\}$, where x represents a user's temporal feature. Each x is a six-tuple $x = \langle ID, TS, LA, LO, SP, DI \rangle$, where ID is the user identifier, TS is the timestamp, and LA, LO, SP, and DI denote the user's latitude, longitude, speed, and direction, respectively.

The user's mobility directly determines the edge server that handles their requests. The next definition specifies the attributes of these edge services, which vary based on the edge server and the service being requested.

Definition 3 (Edge Service Attribute): Given a mobile edge service ecosystem $\mathcal{M} = \langle \mathcal{U}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{R} \rangle$, an edge service is denoted by a tuple $s = \langle ID, CD, SD, BD \rangle$, where $s \in \mathcal{S}$, ID is the unique identifier of the service, and CD, SD, and BD denote the computing, storage, and bandwidth demands of the service, respectively. These values represent the degree of resource demand level of the service for each resource type.

The static properties of edge servers, such as location and resource capacity, are as follows.

Definition 4 (Edge Server Attribute): Given a mobile edge service ecosystem $\mathcal{M} = \langle \mathcal{U}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{R} \rangle$, an edge server is denoted by a tuple $e = \langle ID, LA, LO, RA, CL, SL, BL \rangle$, where $e \in \mathcal{E}$, ID is the server's unique identifier, LA and LO denote its geographical latitude and longitude, RA represents the coverage radius, and CL, SL, and BL denote the computing, storage, and bandwidth capacity levels, respectively, which represent the rated provisioning capacity of the three resource types.

Edge servers are also characterized by dynamic temporal features capturing resource utilization over time.

Definition 5 (Edge Server Temporal Features): In a mobile edge service ecosystem $\mathcal{M}$, each edge server is characterized by its dynamic workload at different timestamps. Let $\mathcal{Y} = \{\mathcal{Y}^1, \mathcal{Y}^2, \ldots, \mathcal{Y}^{|\mathcal{T}|}\}$ denote the collection of edge server feature sets across all timestamps. At a specific time timestamp $t \in \mathcal{T}$, the edge server feature set can be defined as $\mathcal{Y}^t = \{y^t_1, y^t_2, \ldots, y^t_{|\mathcal{E}|}\}$, where y represents the temporal feature of an edge server. Each y can be defined as a five-tuple $y = \langle ID, TS, CR, SR, BR \rangle$, where ID is the identifier

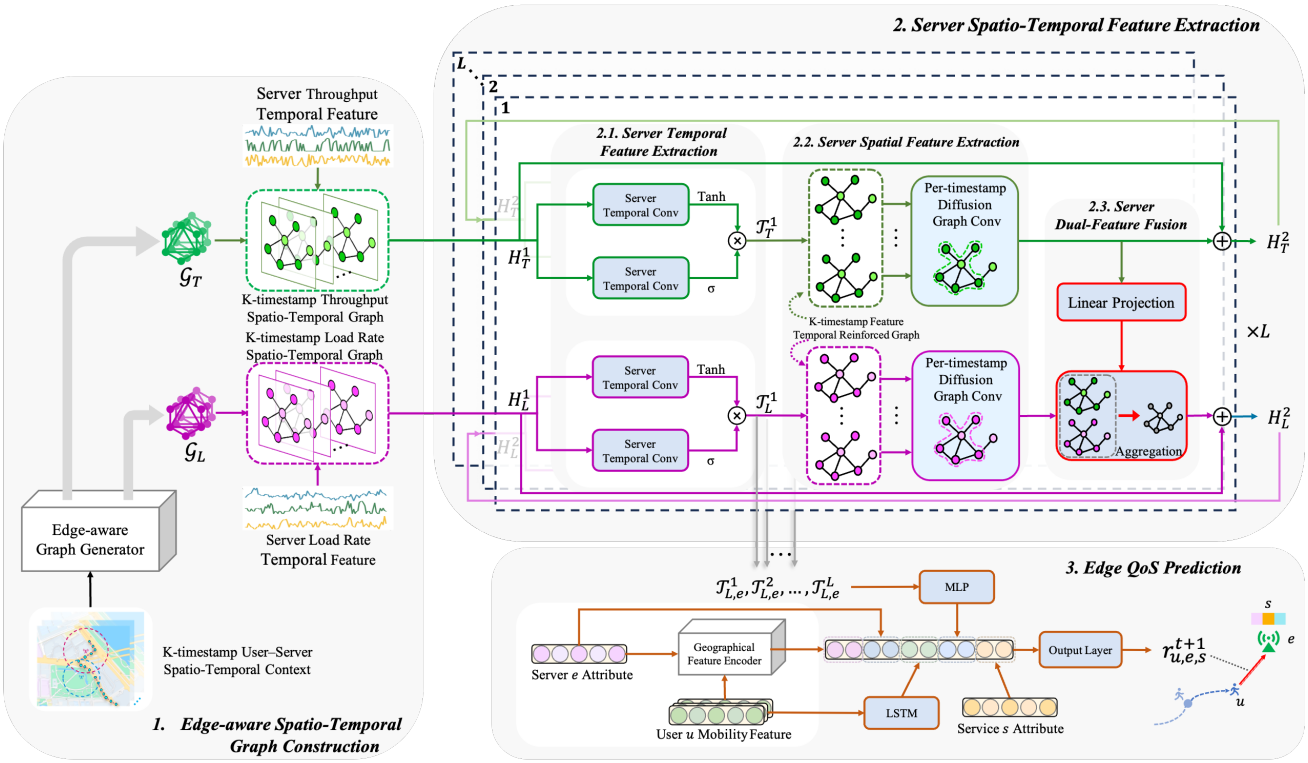


Fig. 1: Overall architecture of the proposed ST-DGL framework for mobile edge QoS prediction.

of the edge server, TS is the timestamp, and CR, SR, and BR denote the computing load rate, storage load rate, and bandwidth load rate of the edge server, respectively, which represent the utilization ratios of the three types of resources at the beginning of each timestamp.

The server throughput represents another essential temporal feature. In this paper, it is used to abstract the total service demand handled by a server at each timestamp, and is formalized as the sum of the computing, storage, and bandwidth demand levels of all service requests handled during that timestamp. These temporal features are essential for predicting QoS. The next definition outlines the formalization of the QoS prediction problem.

Definition 6 (Mobile Edge QoS Prediction Problem): Given the ecosystem of $\mathcal{M}$ up to timestamp t and data from the previous k timestamps, the objective is to predict the QoS value for a specific user u invoking a service s on an edge server e at the next timestamp $t{+}1$. The prediction problem is formalized as follows:

$$\hat{r}^{t+1}_{u,e,s} = \mathcal{F}\big(u, e, s, \mathcal{E}, \mathcal{S}, \mathcal{X}^{\Delta_t^{(k)}}, \mathcal{Y}^{\Delta_t^{(k)}}, \mathcal{R}^{\Delta_t^{(k)}} \,|\, \Theta_{\mathcal{F}}\big) \quad (1)$$

where $\Delta_t^{(k)}$ denotes the time window covering the most recent k consecutive timestamps, i.e., $\Delta_t^{(k)} = [t-k+1, t]$. Let $\mathcal{X}^{\Delta_t^{(k)}}$ and $\mathcal{Y}^{\Delta_t^{(k)}}$ represent the temporal features of all users and edge servers within this window. The recent QoS invocation subset is defined as $\mathcal{R}^{\Delta_t^{(k)}} = \{r^i_{u,e,s} \mid u \in \mathcal{U}, e \in \mathcal{E}, s \in \mathcal{S}, i \in \Delta_t^{(k)}\}$, which includes all observed QoS invocation records for user–server–service pairs across the last k timestamps. $\mathcal{F}(\cdot|\Theta_{\mathcal{F}})$ denotes the prediction framework parameterized by $\Theta_{\mathcal{F}}$, which estimates the QoS value at $t{+}1$ by integrating both entity-level data and the spatio-temporal context from the past k timestamps.

The following sections detail our deep learning framework for capturing spatio-temporal dependencies among edge server workloads.

III. Approach

Figure 1 illustrates the ST-DGL framework, which consists of three key components. Taking historical server features and user-server spatio-temporal context as input, it leverages dynamic graph learning and stacked spatio-temporal extraction layers to capture evolving dependencies, mapping these learned high-level representations to future QoS values:

- **Edge-aware Spatio-Temporal Graph Construction**: This module constructs a learnable Dynamic Windowed Spatio-Temporal Graph (DWSTG) using the user and server location relationships observed over the past k timestamps.
- **Server Spatio-Temporal Feature Extraction**: This module consists of L stacked layers. Each layer uses the graph construction module to obtain a spatio-temporal graph tensor of load rate and throughput over the past k timestamps, where the first dimension indexes time and the remaining two dimensions describe the inter-server graph at each timestamp. The subsequent layer takes the output of the previous layer as its input. Each layer sequentially

executes three core operations: *Server Temporal Feature Extraction* processes both graph sequences to capture dependencies across timestamps. *Server Spatial Feature Extraction* performs DWSTG-based spatial convolutions at every timestamp using the corresponding dynamic graph snapshot to model inter-server spatial correlations. *Server Dual-Feature Fusion* integrates the throughput temporal features into the load rate features, enhancing load rate's spatio-temporal representations.

- **Edge QoS Prediction:** This module encodes user mobility, geographical relationships, server attributes, and service attributes, concatenating them with the MLP-refined L-layer server workload features into a feed-forward network to predict the QoS at timestamp $t{+}1$.

A. Edge-aware Spatio-Temporal Graph Construction

This module captures evolving spatial dependencies among edge servers. In mobile edge environments, continuous user movement triggers frequent service migrations, making server workloads heavily dependent on user mobility patterns. While proximate servers exhibit natural spatio-temporal correlations, user flow directions vary significantly (e.g., rush hours). Furthermore, given the massive daily volume of QoS invocations, edge datasets rarely span sufficient continuous days to model daily periodic patterns. Consequently, we construct dynamic spatio-temporal graphs using mobility-aware server contextual features from the past k timestamps, effectively capturing recent service flow relationships and evolving spatial dependencies between server attributes and real-time user mobility.

Thus, by leveraging these features from the past k timestamps, this module effectively captures the evolving spatial dependencies between server attributes and real-time user mobility within their coverage areas.

As shown in Table I, for a server e and its covered user set U_e^τ at timestamp τ, we extract seven spatial relationships for each user $u_i \in U_e^\tau$. We then aggregate them into a mobility-aware server contextual feature for server e. This is a contextual descriptor of the server rather than an attribute of graph edges, and it is defined as

$$\xi_e^\tau = \Big(\mathrm{AvgPooling}\big([d_{u_i}^{1,\tau}, d_{u_i}^{2,\tau}, \ldots, d_{u_i}^{7,\tau}]\big)_{u_i \in U_e^\tau},\ |U_e^\tau|\Big) \quad (2)$$

where ξ_e^τ represents the mobility-aware server contextual feature of server e at timestamp τ, and $|U_e^\tau|$ denotes the number of users within the coverage of edge server e at timestamp τ.

Having obtained the contextual feature ξ_i^τ for each server at timestamp τ, we can aggregate all server contextual features at the same timestamp τ as

$$E_\tau = \{\xi_1^\tau, \xi_2^\tau, \ldots, \xi_{|\mathcal{E}|}^\tau\} \quad (3)$$

where $|\mathcal{E}|$ denotes the total number of edge servers in the edge system, and $E_\tau \in \mathbb{R}^{|\mathcal{E}|\times 8}$.

The above procedure is applied to every timestamp within the past k timestamps of the prediction window. That is, for a prediction window ending at timestamp t, we compute the contextual feature set:

TABLE I: Seven spatial features describing the relationship between server e and user u_i at timestamp τ. Here, $loc = (\phi, \lambda)$ denotes latitude and longitude, v represents speed, θ is the moving direction in radians, and RA is the server coverage radius.

Feature	Description	Calculation
$d_{u_i,e}^{1,\tau}$	Haversine distance	$\mathrm{Haversine}(loc_{u_i}^\tau, loc_e)$
$d_{u_i,e}^{2,\tau}$	Radial velocity component	$v_{u_i}^\tau \cos(\theta_{u_i}^\tau - d_{u_i,e}^{4,\tau})$
$d_{u_i,e}^{3,\tau}$	Tangential velocity component	$v_{u_i}^\tau \sin(\theta_{u_i}^\tau - d_{u_i,e}^{4,\tau})$
$d_{u_i,e}^{4,\tau}$	Bearing angle	$\mathrm{Bearing}(loc_{u_i}^\tau, loc_e)$
$d_{u_i,e}^{5,\tau}$	Normalized distance	$d_{u_i,e}^{1,\tau}/r_e$
$d_{u_i,e}^{6,\tau}$	Relative moving direction	$\lvert\theta_{u_i}^\tau - d_{u_i,e}^{4,\tau}\rvert$
$d_{u_i,e}^{7,\tau}$	Boundary proximity	$r_e - d_{u_i,e}^{1,\tau}$

$$E = \{E_{t-k+1}, E_{t-k+2}, \ldots, E_t\} \quad (4)$$

where $E \in \mathbb{R}^{k\times|\mathcal{E}|\times 8}$ and it serves as the temporal sequence of mobility-aware server contextual features.

While these contextual features capture local environmental information, they alone are insufficient to characterize inter-server interactions. To model the temporal dependencies and dynamic correlations among servers, we construct a DWSTG based on these features. Concretely, the DWSTG is organized as a 3D spatio-temporal graph tensor whose first dimension corresponds to the k timestamps and whose last two dimensions describe the directed inter-server graph at each timestamp. Temporal dependencies are then captured by the subsequent temporal convolution module.

Specifically, the adjacency weight between server pair (i, j) at timestamp τ is computed by capturing high-order interactions across spatial and temporal dimensions:

$$A'_{\tau,i,j} = \sum_{o=1}^{d_{edg}} \sum_{a=1}^{d_{edg}} \sum_{b=1}^{d_{edg}} E^1_{o,a,b} E^2_{\tau,o} E^3_{\tau,i,a} E^4_{\tau,j,b} \quad (5)$$

where d_{edg} is the server contextual feature embedding dimension, $E^1 \in \mathbb{R}^{d_{edg}\times d_{edg}\times d_{edg}}$ and $E^2 \in \mathbb{R}^{k\times|\mathcal{E}|\times d_{edg}}$ are learnable core and temporal tensors initialized via Xavier uniform distribution $\mathcal{U}(-\sqrt{3/d}, \sqrt{3/d})$ [16] and $E^3, E^4 \in \mathbb{R}^{|\mathcal{E}|\times d_{edg}}$ are spatial factors derived from E.

The dynamic adjacency tensor is obtained through element-wise activation and row-wise normalization:

$$A_{\tau,i,j} = \frac{\exp(\max(0, A'_{\tau,i,j}))}{\sum_{m=1}^{|\mathcal{E}|} \exp(\max(0, A'_{\tau,i,m}))} \quad (6)$$

where $A_{\tau,i,j}$ denotes the directed dynamic connectivity from server i to server j at timestamp τ. Using this formulation with independent sets of learnable parameters, we generate two dynamic adjacency tensors, $A_L, A_T \in \mathbb{R}^{k\times|\mathcal{E}|\times|\mathcal{E}|}$, for the load rate and throughput dynamic graphs ($\mathcal{G}_L$ and $\mathcal{G}_T$). These graphs capture inter-server resource dependency relations induced by user mobility and the resulting service migration, and are subsequently integrated with temporal embeddings to yield the final spatio-temporal server representations.

The load rate sequence is directly extracted from the dataset, recording CR, SR and BR of the servers at each timestamp. The throughput sequence is derived by summing the computing, storage, and bandwidth demand levels of all service requests handled by the server at each timestamp. In this way, throughput is used to abstract the overall amount of service demand handled by the server at that timestamp. $\mathcal{G}_L$ and $\mathcal{G}_T$ are then combined with the temporal embeddings of load rate and throughput to generate the final spatio-temporal representations of edge servers, which include both graph structures and node features.

B. Server Spatio-Temporal Feature Extraction

The spatio-temporal feature extraction module consists of L stacked layers with identical architectures but independent parameters. Each layer performs temporal and spatial feature extraction on the dynamic graphs of load rate and throughput. At the end of each layer, the throughput feature is fused with the load rate to enhance its representation before propagating to the next layer.

1) ***Server Temporal Feature Extraction****:* To capture significant temporal variations in server load and throughput in mobile edge environments, a parallel approach with identical architecture but independent parameters is employed for feature extraction. For each feature, the time series is processed through two independent temporal convolutions and then input into a gated network, where tanh and sigmoid activation functions implement adaptive temporal modulation.

For the l-th layer, the temporal feature tensor $\mathbf{H}_*^l$, where $*$ denotes either L (load rate) or T (throughput), has shape $\mathbb{R}^{k\times|\mathcal{E}|\times d_{st}}$, with d_{st} denoting the feature dimension, and is processed by two server temporal convolutions. Each server temporal convolution is implemented as a depthwise separable convolution [17], consisting of a depthwise convolution and a pointwise convolution. Specifically, the depthwise convolution applies a 2D causal dilated kernel [18] with a layer-dependent dilation rate K_l following the HDC pattern [19] to capture temporal patterns, while the pointwise convolution uses a 3D causal kernel to fuse these extracted features into refined temporal representations. The computation is defined as:

$$\mathbf{Z}^l_{*,(1)} = \mathbf{H}^l_* *^{(1)}_{\text{DW}} \mathbf{D}^l_{*,(1)}, \quad \mathbf{Z}^l_{*,(2)} = \mathbf{H}^l_* *^{(2)}_{\text{DW}} \mathbf{D}^l_{*,(2)} \tag{7a}$$

$$\mathbf{C}^l_{*,(1)} = \mathbf{Z}^l_{*,(1)} *^{(1)}_{\text{PW}} \mathbf{P}^l_{*,(1)}, \quad \mathbf{C}^l_{*,(2)} = \mathbf{Z}^l_{*,(2)} *^{(2)}_{\text{PW}} \mathbf{P}^l_{*,(2)} \tag{7b}$$

$$\mathcal{T}^l_* = \tanh\big(\mathbf{C}^l_{*,(1)}\big) \odot \sigma\big(\mathbf{C}^l_{*,(2)}\big) \tag{7c}$$

In these equations, $\mathbf{H}^l_*$ is processed by two parallel branches, each consisting of a depthwise convolution $*_{\text{DW}}$ with learned parameter matrix $\mathbf{D}^l_*$ and a pointwise convolution $*_{\text{PW}}$ with learned parameter tensor $\mathbf{P}^l_{*,(1)}$ and $\mathbf{P}^l_{*,(2)}$, producing the server temporal convolution outputs $\mathbf{C}^l_{*,(1)}$ and $\mathbf{C}^l_{*,(2)}$. These outputs are activated by $\tanh(\cdot)$ and $\sigma(\cdot)$, respectively, and combined via element-wise multiplication to yield the temporally reinforced feature $\mathcal{T}^l_*$.

2) ***Server Spatial Feature Extraction****:* An edge server's future load rate and throughput depend on the shifting service requests driven by users remaining within its coverage area, alongside those migrating in from or out to spatially related servers across recent timestamps.

Cross-timestamp dependencies are first captured through temporal feature extraction, aggregating information from the past k timestamps into enriched server representations at timestamp t. Spatial feature extraction then propagates these representations using dynamic adjacency matrices A_L and A_T. These graphs encode spatial relationships influenced by user mobility patterns over the past k timestamps. Since the temporal module preserves the k-timestamp dimension, the spatial module operates on k load rate and throughput graphs per layer. Each graph is processed independently by a graph diffusion convolution inspired by DCRNN [20], producing spatially enhanced features passed to the next layer.

To capture mobility-driven bidirectional service flow propagation, we employ a graph-based bidirectional diffusion convolution with layer-independent parameters. Using the temporally extracted load rate ($\mathcal{T}^l_L$) and throughput ($\mathcal{T}^l_T$) at layer l as inputs to the spatial module, the diffusion process is formulated as:

$$\mathcal{X}^l_* = \sum_{k=0}^{p} \Big((D_*^{-1}A_*)^k W^{f,l}_{k,*} + (D_*^{-1}A_*^\top)^k W^{b,l}_{k,*} \Big) \mathcal{T}^l_* \tag{8}$$

where A_* denotes the dynamic adjacency tensor extracted from the tensor A_L or A_T. The diffusion operation is applied independently at each timestamp. The two feature streams share the same diffusion structure but are parameterized independently. Specifically, for each diffusion step $k \in \{0, 1, \ldots, p\}$, separate parameter matrices $\{W^{f,l}_{k,L}, W^{b,l}_{k,L}\}$ and $\{W^{f,l}_{k,T}, W^{b,l}_{k,T}\}$ are learned for the load rate and throughput graphs, respectively.

3) ***Server Dual-Feature Fusion****:* After the spatial feature extraction, the module produces two groups of spatially enhanced temporal features, denoted as $\mathcal{X}^l_L$ for load rate and $\mathcal{X}^l_T$ for throughput, each comprising k historical timestamps.

Since a server's load rate is the primary factor affecting QoS and is strongly influenced by throughput, throughput features are propagated into the load rate graph at each of the k aligned timestamps to enrich its representation. This fusion effectively captures dynamic server workload evolution, thereby enhancing prediction accuracy.

The output of the fusion module at layer l is fed through the residual structure as the input to layer $l{+}1$:

$$\mathbf{H}^{l+1}_L = \mathbf{H}^l_L + F(\mathcal{X}^l_L, \mathcal{X}^l_T) \tag{9a}$$

$$\mathbf{H}^{l+1}_T = \mathbf{H}^l_T + \mathcal{X}^l_T \tag{9b}$$

where $\mathcal{X}^l_L$ and $\mathcal{X}^l_T$ denote the outputs of the l-th layer of spatial feature extraction. Although fusion is performed on each timestamp separately, the tensor form processes them in parallel.

The linear projection maps the throughput features to the load rate feature space:

$$\mathrm{Proj}(\mathcal{X}_T^l) = \mathcal{X}_T^l W \tag{10}$$

where W is a learnable linear transformation shared across all k timestamps, ensuring a consistent mapping into the load rate space.

The fusion function F integrates the projected throughput features with the load rate features:

$$F(\mathcal{X}_L^l, \mathcal{X}_T^l) = \mathcal{X}_L^l + \mathrm{Proj}(\mathcal{X}_T^l) \tag{11}$$

This integrates throughput into the load rate tensor to enhance server feature representations. Stacking L layers iteratively enriches dynamic representations of edge server workloads and service capabilities.

C. Edge QoS Prediction

At timestamp $t{+}1$, QoS depends on the user-server spatial relationship, user mobility, static server/service attributes, and server workload states. To represent the server states, we utilize the load rate features extracted from the L-layer spatio-temporal module, as they inherently incorporate throughput dynamics, providing a comprehensive view of resource availability.

To capture user mobility patterns from historical trajectories, we employ an LSTM network. The mobility feature of user u at timestamp τ is:

$$\mathbf{m}_u^\tau = [\mathbf{p}_u; \phi_u^\tau, \lambda_u^\tau, v_u^\tau, \theta_u^\tau] \tag{12}$$

where $\mathbf{p}_u \in \mathbb{R}^{d_{emb}}$ is the user ID embedding, ϕ_u^τ and λ_u^τ are latitude and longitude, v_u^τ is velocity, and θ_u^τ is direction. The mobility sequence over the past k timestamps is $\mathbf{M}_u = \{\mathbf{m}_u^{t-k+1}, \ldots, \mathbf{m}_u^t\} \in \mathbb{R}^{k\times(d_{emb}+4)}$. An LSTM encoder with hidden dimension d_{tra} processes $\mathbf{M}_u$, using its final hidden state as the user mobility representation $\mathbf{x}_u \in \mathbb{R}^{d_{tra}}$.

For a specific edge server e, its static feature is:

$$\mathbf{m}_e = [\mathbf{p}_e; \mathbf{p}_{CL}; \mathbf{p}_{SL}; \mathbf{p}_{BL}] \tag{13}$$

where $\mathbf{p}_e, \mathbf{p}_{CL}, \mathbf{p}_{SL}, \mathbf{p}_{BL} \in \mathbb{R}^{d_{emb}}$ are embeddings for the server ID, computing level, storage level, and bandwidth level, respectively.

The geographical relationship between u and e at timestamp t is encoded by a two-layer MLP:

$$\mathbf{x}_{\mathrm{geo}} = f_{\mathrm{geo}}\big([d_{u,e},\, \theta_{u,e},\, c_{u,e}]\big) \tag{14}$$

where, at timestamp τ, $d_{u,e}$ is the haversine distance, $\theta_{u,e}$ is the bearing angle, and $c_{u,e}$ is a binary coverage indicator (1 if within radius RA, 0 otherwise).

Next, we collect server e's temporally reinforced load rate vectors from all L layers, $\{\mathcal{T}_{L,e}^1, \mathcal{T}_{L,e}^2, \ldots, \mathcal{T}_{L,e}^L\}$, which capture multiscale temporal dependencies. These are projected into a unified representation across k timestamps:

$$\mathbf{x}_e = \mathrm{Mean}\left([\{\mathcal{T}_{L,e}^1, \mathcal{T}_{L,e}^2, \ldots, \mathcal{T}_{L,e}^L\}]\right) \tag{15}$$

We exclusively use load rate temporal tensors to predict QoS at $t{+}1$. The load rate captures the server's state at the exact start of a timestamp, whereas the throughput spans the entire duration and serves mainly as a complementary indicator of demand. Furthermore, since temporal extraction follows dual-feature fusion, each $\mathcal{T}_L^l$ inherently embeds spatially propagated throughput signals. Finally, to forecast $t{+}1$, we directly output the temporal features to prevent their temporal momentum from being over-smoothed by spatial aggregations.

Finally, the geographical relationship representation $\mathbf{x}_{geo}$, the user mobility feature $\mathbf{x}_u$, the edge server resource representation $\mathbf{x}_e$, the server attribute embedding $\mathbf{m}_e$ and the service attribute embedding $\mathbf{m}_s$ are concatenated as input to a feedforward predictor network:

$$\hat{r}_{u,e,s}^{t+1} = f_{\mathrm{pred}}\left([\mathbf{x}_{geo}; \mathbf{x}_u; \mathbf{x}_e; \mathbf{m}_e; \mathbf{m}_s]\right) \tag{16}$$

where f_{pred} consists of fully connected layers with decreasing neuron counts to regress the QoS value.

The ST-DGL framework is optimized using a mini-batch Mean Squared Error (MSE) loss:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{(u,e,s,t)\in\mathcal{B}} \left(\hat{r}_{u,e,s}^{t+1} - r_{u,e,s}^{t+1}\right)^2 \tag{17}$$

The model is trained for 100 epochs via Adam optimizer, decayed by a factor of 0.1 at milestones $\{20, 50, 80\}$.

TABLE II: STATISTICS OF the CHESTNUT DATASET

Metric	Count
Number of users	1,000
Number of servers	1,763
Number of services	1,000
Server load records	1,269,360
QoS invocation records	33,551,574

TABLE III: Parameter Settings of ST-DGL

Parameter	Value	Description
d_{emb}	8	Feature embedding dimension
d_{tra}	16	User trajectory LSTM dimension
d_{edg}	8	Server contextual feature dimension
d_{st}	32	Edge server feature dimension
d_{geo}	8	Geographical encoding dimension
k	9	Window size
p	2	Diffusion step
L	4	Number of spatio-temporal layers
c	3	Temporal kernel size
K	[1, 2, 1, 2]	Temporal dilation
M	[64, 32, 16, 8, 4, 1]	QoS predictor dims

IV. EXPERIMENTS

A. Experimental Setup and Datasets

All our experiments were conducted on a workstation equipped with an NVIDIA Geforce RTX 4090 GPU and an Intel Xeon Silver 4210 R CPU @ 2.40GHZ. The components of ST-DGL were implemented using Python 3.7.15 and PyTorch 1.13.1.

TABLE IV: Performance Comparisons of Mobile Edge QoS Prediction Among Competing Methods on the Dataset

Methods	Density=5%		Density=10%		Density=15%		Density=20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
GAT	40.52	85.41	44.63	70.84	37.27	61.50	31.45	49.22
LSTM	38.71	73.86	42.48	59.27	34.91	50.32	29.84	42.73
GRU	35.93	70.72	41.57	57.85	33.59	48.41	27.51	40.16
STGCN	32.86	68.93	21.24	55.02	19.12	45.11	18.94	39.70
DCRNN	30.92	66.42	19.75	53.90	17.94	43.45	17.92	39.33
AGCRN	39.58	52.88	17.32	52.08	17.52	42.64	15.71	37.28
MTGNN	28.26	64.93	17.03	47.12	15.01	41.63	14.02	36.82
DSTAGNN	25.95	53.65	18.81	45.31	15.38	42.97	12.53	37.02
MEC-RDESN	61.16	82.94	53.55	74.01	49.42	63.75	45.11	58.67
Pred$_{QoS}$	**19.81**	**50.72**	18.97	47.94	18.03	45.19	17.64	43.78
ST-DGL	20.41	54.83	**14.82**	**41.59**	**13.37**	**37.82**	**10.86**	**34.79**
Gains*	-3.03%	-8.10%	12.98%	8.21%	10.93%	9.15%	13.33%	5.51%

* Gains are computed against the best baselines.

To evaluate the effectiveness of ST-DGL, we have conducted extensive experiments on the CHESTNUT dataset. It is a public benchmark that incorporates real-world user mobility patterns and edge server distributions, sourced from taxi trajectories and telecom base stations in Shanghai. It spans 720 consecutive timestamps, recorded at 30-second intervals, and includes mobility data for 1,000 users, geographical information for 1,763 edge servers, three categories of resource load rate over time, and resource demand profiles for 1,000 edge services. In total, it contains 33,551,574 QoS invocation records, each linking a user, an edge server, a service, a timestamp, and the corresponding RT value. We use CHESTNUT because it is currently the only public QoS prediction dataset specifically oriented to mobile edge environments, and it provides the user mobility and server workload information required by ST-DGL. At present, the lack of additional public datasets also limits evaluation across datasets.

For the observation protocol, we evaluate four QoS densities, namely 5%, 10%, 15%, and 20%, to cover extremely sparse to moderately observed settings. Following common practice in QoS prediction, at each density we randomly sample the corresponding proportion of QoS records as observed entries and use the remaining entries for evaluation. This protocol is particularly important here because mobile edge measurements are often incomplete in practice. The detailed statistics of the experimental dataset are shown in Table II.

The prediction performance of ST-DGL is sensitive to the parameter settings summarized in Table III.

B. Evaluation Metrics

We evaluate ST-DGL using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Let $r^t_{u,e,s}$ and $\hat{r}^t_{u,e,s}$ denote the actual and predicted QoS for user u invoking service s via edge server e at timestamp t, defined as:

$$\mathrm{MAE} = \frac{1}{N}\sum_{u,e,s}\left|r^t_{u,e,s} - \hat{r}^t_{u,e,s}\right| \tag{18}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{u,e,s}\left(r^t_{u,e,s} - \hat{r}^t_{u,e,s}\right)^2} \tag{19}$$

where N is the total number of test samples. Lower values indicate higher prediction accuracy. While MAE intuitively reflects the overall average error, RMSE heavily penalizes large deviations, making it particularly effective for evaluating robustness against abrupt QoS fluctuations caused by rapid user movement or server workload shifts in dynamic edge environments.

C. Competing Methods

ST-DGL is evaluated against ten baselines spanning multiple categories. Their details are as follows:

- *GAT [21]*: It employs self-attention mechanisms to weigh neighboring nodes, adaptively aggregating spatial information from edge servers.
- *LSTM [22]*: It utilizes memory cells and gating mechanisms to model long-term temporal dependencies in sequential data.
- *GRU [23]*: It captures temporal dependencies via reset and update gates, providing highly efficient sequence modeling with rapid convergence.
- *STGCN [24]*: It integrates temporal and spatial graph convolutions to jointly capture spatial correlations and temporal dynamics among edge servers.
- *DCRNN [20]*: It combines diffusion graph convolution with recurrent units to jointly model spatial dependencies and temporal evolution over directed graphs.
- *AGCRN [25]*: It introduces learnable node-specific embeddings to autonomously construct a dynamic, data-

TABLE V: Ablation Study Results on the Dataset

Density = 5%			**Density = 15%**		
Methods	MAE	RMSE	Methods	MAE	RMSE
ST-DGL-DF	43.15 (+52.7%)	72.84 (+24.7%)	ST-DGL-DF	18.55 (+27.9%)	53.70 (+29.6%)
ST-DGL-F	36.35 (+43.9%)	65.33 (+16.1%)	ST-DGL-F	16.85 (+20.7%)	45.05 (+16.0%)
ST-DGL-D	32.60 (+37.4%)	68.42 (+19.9%)	ST-DGL-D	17.25 (+22.5%)	49.10 (+22.9%)
ST-DGL	**20.41**	**54.83**	**ST-DGL**	**13.37**	**37.82**
Density = 10%			**Density = 20%**		
Methods	MAE	RMSE	Methods	MAE	RMSE
ST-DGL-DF	29.46 (+49.7%)	57.13 (+27.2%)	ST-DGL-DF	16.28 (+33.3%)	47.95 (+27.5%)
ST-DGL-F	23.84 (+37.8%)	51.94 (+19.9%)	ST-DGL-F	14.91 (+27.2%)	41.60 (+16.4%)
ST-DGL-D	25.12 (+41.0%)	53.82 (+22.7%)	ST-DGL-D	15.42 (+29.6%)	44.52 (+21.9%)
ST-DGL	**14.82**	**41.59**	**ST-DGL**	**10.86**	**34.79**

driven graph structure for enhanced spatial-temporal representation learning.

- *MTGNN [26]*: It couples temporal convolution with adaptive graph learning to dynamically model long-range dependencies and evolving spatial relationships.
- *DSTAGNN [27]*: It leverages attention-based temporal encoding alongside adaptive adjacency learning to model time-varying spatial dependencies.
- *MEC-RDESN [14]*: It integrates user mobility awareness with a dynamic Echo State Network (ESN) for swift mobile edge QoS prediction.
- *Pred$_{QoS}$ [13]*: It employs VQ-VAE to compress data into discrete binary codes, explicitly capturing temporal dynamics to forecast continuous QoS sequences.

We first construct a predefined static graph $\mathcal{G}_S$ based on server geographic locations using DBSCAN [28]. For a controlled comparison, baseline models replace only our core spatio-temporal module. Specifically, temporal models focus exclusively on time-series extraction of server workload features without spatial consideration. Spatial models utilize server features after spatial propagation from neighbors at the previous timestamp via $\mathcal{G}_S$. Spatio-temporal baselines substitute our module by utilizing their internal graph modeling, using either feature-based graph generation or the provided $\mathcal{G}_S$, to capture dependencies among server workloads.

D. Experiment Results and Analyses

As shown in Table IV, baseline models exhibit distinct performance. GAT yields relatively high errors across all densities ; despite adaptively aggregating spatial information, its reliance on static spatial dependencies without temporal awareness fails to capture rapid mobility-induced dynamics. Conversely, while LSTM and GRU improve accuracy by modeling historical temporal dependencies , their inability to exploit explicit server topological information inherently limits their capability to capture cross-server interactions, which reduces their adaptability and robustness in edge scenarios with frequent service request handovers.

Spatio-temporal models (STGCN, DCRNN, MTGNN, AGCRN, DSTAGNN) jointly capture spatial and temporal dependencies. STGCN and DCRNN rely on a static graph $\mathcal{G}_S$, limiting adaptability, while AGCRN, MTGNN, and DSTAGNN introduce adaptive graph construction or attention mechanisms. Nevertheless, all overlook broader mobile edge contexts such as mobility-driven interactions, constraining their suitability for edge QoS prediction.

MEC-RDESN employs an Echo State Network to integrate mobility awareness for QoS prediction. However, its reliance on static server attributes neglects essential server-level dynamic attributes, such as workload fluctuations. This limits its ability to capture complex inter-server dependencies. In contrast, Pred$_{QoS}$ utilizes VQ-VAE to model temporal patterns, performing strongly at low data densities (e.g., 5%). Under such an extremely sparse setting, its discrete hash codes help map sparse data to relatively stable latent representations, so ST-DGL remains competitive rather than dominant. However, as density increases, the coarse static hashing and lack of mobile edge context cause performance gains to plateau. Consequently, Pred$_{QoS}$ faces diminishing returns and increased overfitting risks at higher densities. Therefore, neither method delivers superior prediction performance.

Addressing these limitations, ST-DGL constructs dynamic spatio-temporal graphs from recent history to jointly model server load rate and throughput, adaptively capturing mobility and load dynamics. Consequently, ST-DGL consistently achieves lower MAE and RMSE across all densities, demon-

strating superior robustness in dynamic edge environments.

E. Ablation Study

To evaluate the effectiveness of the proposed ST-DGL framework, we conducted comprehensive ablation studies focusing on its two principal components, namely the DWSTG and the Feature Fusion Module (FFM). Specifically, we compare the full model against four experimental variants:

- *ST-DGL-DF group:* The reference variant excluding both DWSTG and FFM, relying solely on primary features and the static graph $\mathcal{G}_S$.
- *ST-DGL-F group:* Employs only DWSTG, learning load rate features independently without throughput guidance.
- *ST-DGL-D group:* Utilizes only FFM, fusing throughput impacts onto load rates using the $\mathcal{G}_S$.
- *ST-DGL group:* The complete model integrating both components for joint modeling of dynamic relationships and dual feature interactions.

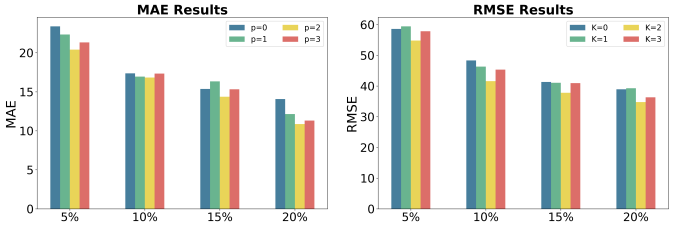
We evaluated ST-DGL across varying data densities (5%, 10%, 15%, and 20%). As Table V shows, ST-DGL consistently outperforms its variants. Component analysis reveals both DWSTG and FFM independently enhance predictions, since ST-DGL-F and ST-DGL-D surpass ST-DGL-DF.

Comparing the variants further reveals a density-dependent tradeoff. At the extremely sparse 5% setting, ST-DGL-D is slightly stronger than ST-DGL-F, indicating that direct workload-feature fusion is particularly helpful when observations are scarce. In contrast, at 10%–20% densities, ST-DGL-F consistently outperforms ST-DGL-D, showing that dynamic graph modeling becomes more effective once sufficient observations are available to recover richer mobility-driven inter-server relations. Consequently, achieving accurate QoS prediction in highly dynamic edge environments demands the combination of contextual graph modeling and workload feature fusion.

F. Performance Impact of Parameters

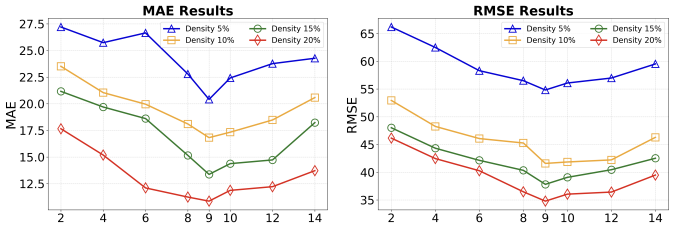
*1) **Impact of Diffusion Step**:* The diffusion step p determines the spatial receptive field by defining the number of aggregation hops. We evaluated $p \in \{0, 1, 2, 3\}$ across four data densities. As Figure 2 illustrates, increasing p generally improves MAE and RMSE. This gain is most pronounced under low data densities, where $p = 2$ performs best by effectively mitigating sparsity. Conversely, as density increases, performance across p values converges, indicating our architecture efficiently captures deeper spatial dependencies. However, larger diffusion steps degrade accuracy. Since distant servers may cross regional workload boundaries, excessive propagation can aggregate irrelevant features and cause over-smoothing.

*2) **Impact of Window Size**:* In ST-DGL, the window size k dictates the number of recent timestamps used to construct the DWSTG and extract temporal features. To evaluate its impact, we fixed the diffusion step to $p = 2$ and varied k across four data densities (5%, 10%, 15%, and 20%). As Figure 3 illustrates, the model achieves optimal performance



(a) MAE of various p (b) RMSE of various p

Fig. 2: Performance impact of diffusion step.



(a) MAE of various k (b) RMSE of various k

Fig. 3: Performance impact of window size.

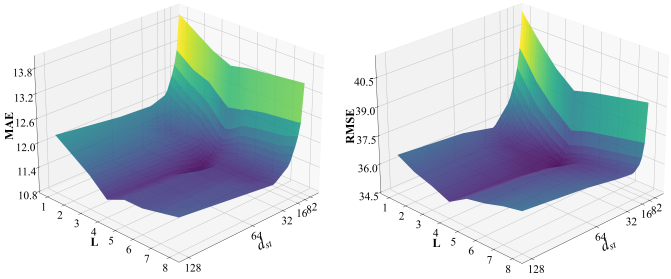
at $k = 9$, where both MAE and RMSE reach their minimum values. Windows smaller than this duration fail to accumulate sufficient trajectory momentum to predict handover trends, while excessively large windows integrate stale states of users who have long migrated to distant base stations. Consequently, $k = 9$ offers the most balanced temporal receptive field, providing sufficient recent behavioral information while avoiding historical noise.

*3) **Impact of Spatio-Temporal Layers and Feature Dimension**:* The layers L and feature dimension d_{st} jointly govern the model's capacity to capture complex dependencies. Specifically, L determines extraction depth via temporal convolutions using a fixed kernel ($c = 3$) and an HDC dilation pattern [19] $[1, 2, 1, 2, ...]$ to expand the temporal receptive field, while d_{st} controls representation expressiveness. To evaluate their interaction, we varied $L \in [1, 8]$ and $d_{st} \in [2, 128]$ under optimal settings ($k = 9, p = 2$, 20% density). As Figure 4 illustrates, the performance surface reveals a distinct optimum at $L = 4$ and $d_{st} = 32$, achieving an MAE of 10.86 and RMSE of 34.79. Although performance remains strong across intermediate d_{st} ranges, increasing d_{st} to 128 consistently degrades accuracy across all layer depths due to overfitting and computational complexity.

V. Related Work

A. Static QoS Prediction

Static QoS prediction methods focus on modeling user–service interactions from historical records without explicitly considering temporal variations, primarily utilizing collaborative filtering or deep learning. Early works predominantly relied on memory-and model-based collaborative filtering to uncover latent patterns, using techniques such as entity sim-



(a) MAE of various L & d_{st} (b) RMSE of various L & d_{st}

Fig. 4: Performance impact of spatio-temporal layers and feature dimension.

ilarity for neighbor selection [11] and location-aware matrix factorization incorporating user reputation [29].

However, since traditional linear models struggle with complex non-linear interactions, deep learning methods have been adopted for their strong feature extraction capabilities. Yin et al. [30] proposed a deep framework extracting latent representations from multiple data sources, while Zou et al. [31] introduced Neighborhood-based Collaborative Residual Learning (NCRL) to learn discriminative representations. More recently, Wu et al. [32] and Liu et al. [9] leveraged graph neural networks to model user–service relationships using reputation information and high-order connectivity patterns, respectively. Despite successfully capturing spatial and structural dependencies, these static models fundamentally neglect temporal dynamics and long-term trends, limiting their accuracy in real-world scenarios where QoS fluctuates over time.

B. Time-aware QoS Prediction

To address the limitations of static models, time-aware QoS prediction approaches have been developed to capture fluctuations by explicitly incorporating time factors. Early efforts focused on extending factorization techniques to the temporal dimension. For instance, Chen et al. [33] presented a temporal regularized tensor factorization mechanism to capture the latent features in time-series QoS data for accurate prediction.

Along this line, Keshavarzi et al. [34] addressed time-series QoS prediction by clustering temporal datasets using an enhanced k-medoids algorithm prior to collaborative filtering, thereby mitigating cold-start problems through dynamic time warping techniques. Subsequently, Tong et al. [35] proposed a time-aware collaborative filtering method that partitions QoS data into time slices for neighbor selection to alleviate short-term timeliness decay. Building on these temporal modeling efforts, Zou et al. [10] further advanced time-aware QoS prediction by proposing TRCF, which exploits continuous temporal QoS vectors across multiple time slices to model temporal fluctuations while alleviating data sparsity. Although these methods effectively model temporal fluctuations, they often fail to fully adapt to the unique constraints of mobile edge environments, particularly the dynamic dependencies driven by user mobility and server workload variations.

C. Edge QoS Prediction

Despite the progress in temporal modeling, the aforementioned methods often fail to fully adapt to the unique constraints of mobile edge environments. Early studies mainly extended existing models without explicitly modeling edge-specific dynamics. Yin et al. [30] proposed a deep feature learning framework using heterogeneous contextual information, while Choi et al. [36] designed an edge-oriented model based on node-service relationships. However, these approaches largely rely on static or weakly dynamic representations and lack explicit server contextual factors. More recently, Jin et al. [14] developed MEC-RDESN, utilizing an Echo State Network for swift, mobility-aware QoS forecasting. However, this framework relies heavily on user features, neglecting essential server-level dynamic properties. Additionally, Zhang et al. [13] evaluated a VQ-VAE-based time-aware model on the CHESTNUT dataset, indicating that advanced temporal techniques can generalize to edge data, but still lack explicit modeling of server workload dynamics.

While previous approaches, including recent advancements like MEC-RDESN and Pred$_{QoS}$, have made significant progress, they still fall short in fully capturing the dynamic nature of edge environments, particularly fluctuating server resource availability. These methods also lack effective adaptation to different time periods, failing to account for complex temporal fluctuations in server load and throughput. In contrast, our approach, ST-DGL, introduces dynamic spatio-temporal attributes of edge servers to explicitly model their spatial and temporal dependencies. By leveraging neighboring timestamps, user mobility, and server workload fluctuations, ST-DGL effectively captures these dynamics, laying the foundation for applying spatio-temporal models to QoS prediction in mobile edge environments.

VI. Conclusion and Future Work

In this paper, we propose ST-DGL for mobile edge QoS prediction to capture server workload dynamics and time-varying mobility patterns. Specifically, to model complex fluctuations in server states, we design the FFM to capture workload dynamics by dynamically integrating load rate and throughput. Furthermore, to address shifting spatial dependencies from varying user flows, we introduce the DWSTG, which organizes the recent k timestamps as a spatio-temporal graph tensor to adaptively learn evolving inter-server relationships driven by real-time mobility. Experimental results demonstrate that ST-DGL significantly outperforms existing methods, validating its effectiveness in dynamic edge environments.

Future work will refine these spatio-temporal modeling techniques to better adapt to diverse user mobility and dynamic edge workloads. As more public mobile edge QoS datasets become available, we will also evaluate the transferability of ST-DGL across different datasets.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62272290, 62572114).

REFERENCES

[1] G. Zou, F. Zhao, and S. Hu, "Chestnut: A qos dataset for mobile edge environments," 2024.

[2] D. Kostadimas, V. Kasapakis, and K. Kotis, "A systematic review on the combination of VR, IoT and AI technologies, and their integration in applications," *Future Internet*, vol. 17, no. 4, p. 163, 2025.

[3] M. A. Khan, E. Baccour, Z. Chkirbene, A. Erbad, R. Hamila, M. Hamdi, and M. Gabbouj, "A survey on mobile edge computing for video streaming: Opportunities and challenges," *IEEE Access*, vol. 10, pp. 120 514–120 550, 2022.

[4] Y. Chen, D. Wang, N. Wu, and Z. Xiang, "Mobility-aware edge server placement for mobile edge computing," *Comput. Commun.*, vol. 208, pp. 136–146, 2023.

[5] H. Sfaxi, I. Lahyani, S. Yangui, and M. Torjmen, "Latency-aware and proactive service placement for edge computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 4, pp. 4243–4254, 2024.

[6] G. Khababa, S. Bessou, F. Seghir, N. H. Harun, A. S. Almazyad, P. Jangir, and A. W. Mohamed, "Collaborative filtering techniques for predicting web service qos values in static and dynamic environments: A systematic and thorough analysis," *IEEE Access*, vol. 13, pp. 45 350–45 376, 2025.

[7] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. Hung, "Personalized LSTM based matrix factorization for online QoS prediction," in *IEEE Int. Conf. Web Serv. (ICWS)*, 2018, pp. 34–41.

[8] S. Fu, J. Li, and L. Wang, "Ra-qos: a robust autoencoder-based qos predictor for highly accurate web service qos prediction," *PeerJ Comput. Sci.*, vol. 11, p. e2928, 2025.

[9] 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, 2023.

[10] 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, 2023.

[11] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 134–144, 2019.

[12] Z. Liu, Q. Z. Sheng, X. Xu, D. Chu, and W. E. Zhang, "Context-aware and adaptive QoS prediction for mobile edge computing services," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 400–413, 2019.

[13] L. Kong, X. Hu, L. Qi, X. Xu, Y. Zhang, L. Yao, and X. Zhang, "Deep learning to hash for time-aware QoS prediction based on VQ-VAE," *IEEE Trans. Serv. Comput.*, 2025.

[14] H. Jin, P. Zhang, H. Dong, A. Bouguettaya, and A. Y. Zomaya, "Swift and accurate mobility-aware QoS forecasting for mobile edge environments," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 4340–4353, 2024.

[15] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient parallel split learning over resource-constrained wireless edge networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 10, pp. 9224–9239, 2024.

[16] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2010, pp. 249–256.

[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1251–1258.

[18] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[19] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell, "Understanding convolution for semantic segmentation," in *WACV*. IEEE Computer Society, 2018, pp. 1451–1460.

[20] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[22] A. Graves, "Long Short-Term Memory," in *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence, 2012, vol. 385, pp. 37–45.

[23] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[24] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[25] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *NeurIPS*, 2020.

[26] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *KDD*. ACM, 2020, pp. 753–763.

[27] S. Lan, Y. Ma, W. Huang, W. Wang, H. Yang, and P. Li, "DSTAGNN: dynamic spatial-temporal aware graph neural network for traffic flow forecasting," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 11 906–11 917.

[28] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*. AAAI Press, 1996, pp. 226–231.

[29] S. Li, J. Wen, and X. Wang, "From reputation perspective: A hybrid matrix factorization for QoS prediction in location-aware mobile service recommendation system," *Mob. Inf. Syst.*, vol. 2019, pp. 8 950 508:1–8 950 508:12, 2019.

[30] 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.

[31] 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, 2022.

[32] 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.

[33] Y. Chen, Y. Zhang, H. Xia, C. Gao, Z. Wang, F. Wang, and G. Li, "A hybrid tensor factorization approach for QoS prediction in time-aware mobile edge computing," *Appl. Intell.*, vol. 52, no. 7, pp. 8056–8072, 2022.

[34] A. Keshavarzi, A. T. Haghighat, and M. Bohlouli, "Enhanced time-aware QoS prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (QoPC)," *Computing*, vol. 102, no. 4, pp. 923–949, 2020.

[35] 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.

[36] J. Choi, J. Lee, D. Ryu, S. Kim, and J. Baik, "GAIN-QoS: A novel QoS prediction model for edge computing," *J. Web Eng.*, vol. 21, no. 1, 2022.