Journal of Computer Technology and Software

ISSN: 2998-2383

Vol. 5, No. 2, 2026

Container-Level Latency Prediction via Integrated Structure-Aware Graph Modeling and Multi-Scale Temporal Encoding

Feng Chen
Northeastern University, Seattle, USA
ffeng.chen1@gmail.com

Abstract: This paper proposes a container-level latency prediction framework that integrates structure-aware graph modeling and multi-scale temporal modeling to address the challenges of frequent latency fluctuations and complex structural dependencies in containerized backend systems. The framework first introduces a structure-aware graph modeling module that dynamically constructs interaction dependency graphs among containers. It uses graph neural networks to capture structural information in key paths and asymmetric communication chains, enhancing the model's ability to represent causal clues in latency modeling. A multi-scale temporal encoding mechanism is then applied to extract short-term variations and long-term trends across multiple temporal granularities. A learnable convolutional kernel fusion strategy is used to improve robustness in modeling non-stationary latency sequences. During feature integration, structural graph embeddings and multi-scale temporal representations are jointly mapped into the latency prediction space. A unified optimization loss function enforces both cross-scale consistency and structural preservation constraints. To further validate the effectiveness of the modeling components, two sets of key sub-experiments are designed to evaluate the impact of multi-scale fusion strategies and time window sliding steps on latency prediction performance. Experiments conducted on real-world container monitoring datasets show that the proposed model significantly outperforms existing methods across multiple evaluation metrics, demonstrating strong generalization and sensitivity to structural dependencies.

Keywords: Structure-aware graph modeling; multi-scale temporal coding; container delay modeling; joint feature optimizatio

1. Introduction

With the widespread adoption of containerization and microservice architectures in cloud-native systems, container-level network and computation behaviors have become critical factors affecting system performance stability. Due to highly dynamic communication topologies, complex service dependency chains, and unpredictable load fluctuations among containers, achieving accurate modeling and temporal prediction of container latency has become an important research topic. This is essential for improving system resilience and optimizing resource scheduling. In high-concurrency scenarios, the propagation of latency fluctuations may lead to system-level performance bottlenecks or cascading anomalies. Therefore, structured and temporal modeling of container-level behaviors holds significant engineering and theoretical value[1,2].

However, existing mainstream approaches still face many challenges in addressing these problems. Traditional methods based on metric sequences ignore the semantic dependencies between upstream and downstream containers, making it difficult to capture behavior correlations induced by structural relationships. Some graph neural networks attempt to incorporate topological information, but they cannot often model high-order dynamic interactions between nodes, resulting in limited structural expressiveness[3]. Moreover, current temporal modeling techniques typically use single-scale or fixed receptive mechanisms, which cannot effectively capture cross-period variations and short-term spikes in latency patterns. This limits their adaptability and generalization in complex production environments[4].

To address these challenges, this paper proposes a container-level latency modeling method that integrates structure-aware graph modeling and multi-scale temporal encoding. The goal is to jointly explore deep patterns in system structural correlations and temporal behavior evolution. The method introduces a structure-aware graph modeling module to capture container interaction dependencies and propagation paths. A multi-scale convolutional temporal encoding module is then used to jointly model latency patterns across different time granularities. Together, these components enable accurate representation of key behaviors in complex dynamic systems[5].

The main contributions of this study are as follows. First, a Structure-Aware Graph Modeling (SAGM) mechanism is proposed to dynamically construct a semantic interaction graph among containers[6]. A multi-layer graph message passing mechanism is employed to extract structural representations and enhance the modeling of global system dependencies. Second, a Multi-Scale Temporal Encoding (MSTE) module is designed to extract both short-term fluctuations and long-term trends using different receptive windows. This improves the model's ability to represent non-stationary latency sequences. Third, a joint optimization objective is constructed by

combining structural contrastive learning and temporal reconstruction constraints. This facilitates consistent representation across structural and temporal modules and enhances the interpretability and robustness of latency prediction in containerized systems[7].

2. Related work

2.1 Container-Level Latency Modeling and Prediction

With the increasing adoption of containerized infrastructure, network latency has become a critical factor limiting system performance optimization and service quality assurance. Compared with traditional virtual machine architectures, containers offer advantages such as fast startup, low resource overhead, and flexible deployment. However, they are also more vulnerable to host resource contention, dynamic service migration, and changes in network topology. These factors lead to stronger instability and time-varying characteristics in network latency. In microservice architectures in particular, a single user request often requires collaboration across multiple container instances. The service chain is complex, and communications are frequent. As a result, end-to-end latency exhibits strong coupling and multi-hop dependency. In such high-concurrency and heterogeneous environments, simple rule-based thresholds or static indicators are no longer sufficient for effective latency prediction[8,9].

Traditional methods mainly adopt statistical time series modeling techniques to forecast latency trends. These approaches rely on historical observations to identify patterns and infer trends. They can capture periodic fluctuations and long-term variations to some extent. However, the runtime state of container systems is constrained by multi-dimensional resources, dynamic scheduling strategies, and interference from multi-tenant environments[10]. The resulting nonlinearity and burstiness go far beyond the modeling capabilities of linear models. In addition, container runtime data is typically high-dimensional and heterogeneous. Examples include CPU usage, memory consumption, network I/O, and disk operations. Traditional models struggle to extract relevant features and represent system states, making it difficult to uncover the underlying generation mechanism of latency. This often results in predictions that deviate from reality.

With the development of data-driven techniques, deep learning methods have been gradually introduced into latency prediction tasks. Recurrent neural networks, in particular, have shown potential in modeling nonlinear dynamics in time series. Structures such as LSTM and GRU can capture long-term dependencies and non-stationary patterns in latency sequences. This improves prediction accuracy to some degree. However, these models generally ignore the structural dependencies between containers. They treat the system as multiple independent time series tasks. This fails to capture the joint behavior and invocation topology among containers. In microservice systems, interactions between upstream and downstream containers have a significant impact on latency evolution. If such structural factors are not considered, model performance tends to deteriorate rapidly in complex scenarios[11,12].

To address this issue, some studies have incorporated multi-source monitoring indicators and tracing data to enrich the input feature space. These approaches improve context awareness and provide more comprehensive modeling of system states. Nevertheless, most of them still focus on temporal modeling. The representation and use of structural information remain limited. Furthermore, container instances have short lifespans and flexible deployment policies. Their behavior patterns are highly dynamic and difficult to model. Modeling methods that rely on fixed rules or static feature sets cannot adapt to structural drift and state transitions during system evolution. Therefore, how to incorporate structural dependencies among containers into latency modeling and effectively integrate them with temporal dynamics has become a critical challenge in container-level latency prediction research[13].

2.2 Spatiotemporal Modeling in Distributed Systems

In complex distributed systems, system states are often influenced by both temporal evolution and spatial structure. Their dynamic and high-dimensional characteristics make it difficult for traditional modeling methods to capture underlying patterns effectively. Distributed architectures, characterized by multi-node communication and heterogeneous service collaboration, exhibit strong structural coupling and temporal dependencies during runtime. The state of each node changes over time, and the interaction structure continuously evolves. Examples include network topology reconfiguration, task scheduling adjustments, and load redistribution[14]. These factors have a profound impact on latency, resource usage, and behavioral patterns. In this context, single-dimensional modeling approaches often face limitations in expressiveness and fail to capture complex behaviors under multi-factor interactions. Therefore, modeling strategies that integrate both temporal and structural perspectives have become essential for the accurate representation of distributed systems[15].

The core idea of spatiotemporal modeling is to characterize both the temporal evolution of data and the structural dependencies between components. The temporal dimension captures how system states change over time. The spatial dimension describes relationships, interactions, and coupling among entities in the system. To address this goal, many recent studies have proposed graph-based spatiotemporal modeling approaches. Graph neural networks, in particular, have been widely applied to structural modeling and provide strong support for capturing complex dependencies. Each node in the system can be viewed as an individual unit. Edges between nodes represent explicit or implicit interactions. Graph structures can dynamically express service paths, communication dependencies, and topology changes. At the same time, temporal modeling mechanisms such as sliding windows, temporal convolutions, and recurrent networks can capture historical behavior patterns and dynamic trends. This enables joint modeling of complex spatiotemporal behaviors[16,17].

In containerized or microservice-based architectures, this spatiotemporal modeling strategy shows great potential. Communication between containers can be abstracted as a structural graph. Service invocation paths and upstream-

downstream dependencies form a semantic information propagation network. Each container's state evolves. Time series metrics such as load, response latency, and CPU utilization carry key dynamic behavior signals. By modeling containers as graph nodes and communication links as edges, and combining this with temporal modeling to track each node's state, the model can simultaneously capture structural dependencies and dynamic behaviors within a unified framework[18]. This approach addresses the limitations of traditional models that ignore structure or fail to adapt to dynamic changes. It significantly improves the system's behavior representation capacity.

Moreover, with the advancement of deep learning in system modeling, more hybrid models have been proposed to enhance cross-dimensional feature fusion in spatiotemporal learning. Some methods use multi-scale graph modeling to encode system structures hierarchically, capturing dependencies from both global and local perspectives. Others introduce structural attention mechanisms and temporal gating units to emphasize critical behavioral changes of key nodes. More recently, some works have adopted dynamic graph structures that allow models to adjust structural representations during system changes. This increases robustness and adaptability. These studies suggest that in complex and dynamic container environments, spatiotemporal hybrid modeling not only offers stronger representational power but also provides a practical and theoretical foundation for container-level network latency prediction[19,20].

3. Method

To address the challenges of strong temporal dynamics and complex structural dependencies in container-level network latency prediction, this paper proposes a prediction framework based on spatiotemporal hybrid modeling. The framework aims to jointly capture the temporal evolution of container runtime states and the spatial coupling of cross-service communication structures. It includes two core innovations. First, a Structure-Aware Graph Modeling (SAGM) module is designed to construct dynamic communication dependency graphs, enabling precise modeling of upstream and downstream interactions and potential dependency paths between containers. Second, a Multi-Scale Temporal Encoding (MSTE) module is introduced, which uses multi-granularity window convolutions and hierarchical temporal embeddings to represent container state transitions across different time scales. This enables fine-grained modeling of latency fluctuation trends and long-term dependencies. The integration of temporal and structural modeling equips the framework with the ability to perceive both system structure dynamics and local temporal patterns, providing a solid foundation for accurate latency prediction in complex containerized environments. The detailed structure of the proposed model is illustrated in Figure 1.

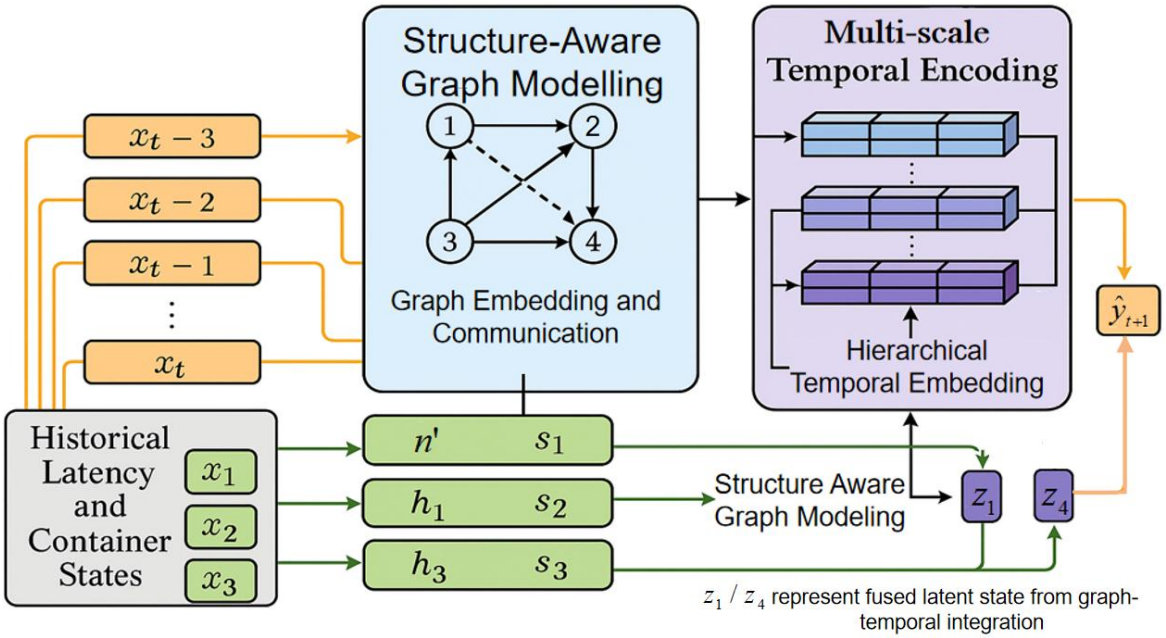

Figure 1. Overall model architecture diagram

3.1 Structure-Aware Graph Modeling

This study presents a Structure-Aware Graph Modeling (SAGM) mechanism aimed at effectively capturing the intricate structural relationships among containers from multiple perspectives, including communication pathways, service-level dependencies, and synchronized behavioral patterns. Unlike traditional modeling approaches that primarily depend on static configurations or isolated performance indicators, SAGM addresses the critical need to model the dynamic and evolving dependency graphs that naturally emerge during containerized system operation. These traditional methods often fall short in recognizing the deep semantic interactions and complex correlations that span across containers during runtime. In contrast, SAGM is

specifically designed to construct richer and more representative structural embeddings that reflect both the functional and temporal interdependencies within distributed systems. By doing so, it lays the foundation for more accurate and robust spatiotemporal representations, which are essential for downstream analytical and predictive tasks. The complete architectural layout of the proposed model is depicted in Figure 2.

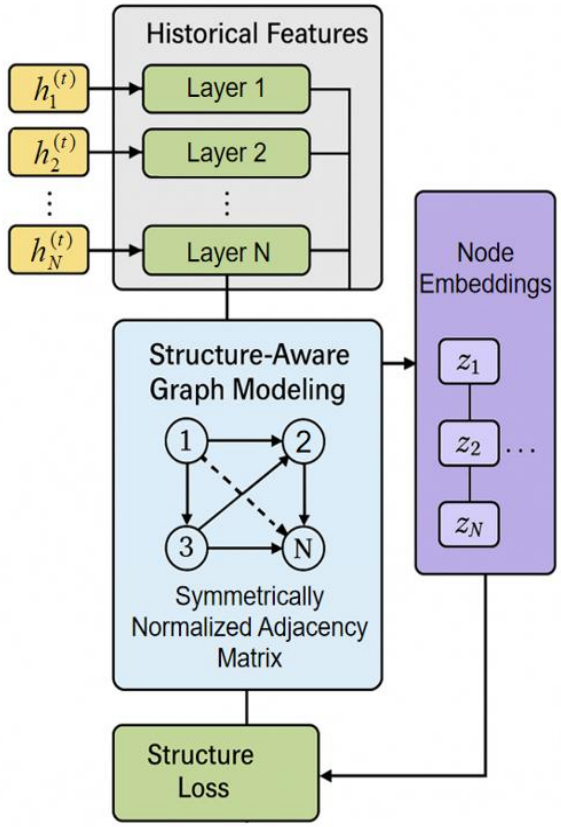

Figure 2. SAGM module architecture

In this mechanism, a dynamic graph construction strategy is first introduced to continuously discover and update the interactions among containers within different time windows. This ensures that the generated graph structure accurately reflects the evolving system state. The graph includes not only explicit invocation paths and dependency chains but also implicit collaborative behavior patterns extracted from temporal logs. Through a node feature propagation mechanism, the system aggregates higher-order structural context within the graph. This enhances the discriminability and identifiability of individual container nodes in the overall representation.

The resulting structure-aware representation plays a critical role in the entire modeling framework. It formalizes multi-granularity dependencies among containers and provides a context-rich input foundation for the subsequent spatiotemporal modeling module. With this mechanism, the model can detect structural disturbances, capture potential interaction patterns, and improve its capacity to represent behavioral deviations with higher modeling accuracy in complex and heterogeneous distributed environments. This module works in coordination with the temporal modeling component to construct a high-dimensional embedding space for container-level behavior modeling.

Assuming that the system consists of N containers at a time step t, a dynamic graph $G_t = (V_t, \varepsilon_t)$ can be constructed, where V_t represents the set of container nodes and ε_t represents the set of edges derived based on communication frequency or service topology. Each node $i \in V_t$ corresponds to an initial feature vector $h_i^t \in R^d$, and the adjacency relationship in the graph is represented by the weighted adjacency matrix $A_t \in R^{N \times N}$. Based on the idea of structure perception, the graph embedding propagation mechanism designed in this paper is as follows:

$$\widetilde{A}_t = D_t^{-\frac{1}{2}} A_t D_t^{-\frac{1}{2}}$$

$$H_t^{(l+1)} = \sigma(\widetilde{A}_t H_t^{(l)} W^{(l)})$$

Among them, D_t is the degree matrix of A_t, $H_t^{(l)}$ represents the l-th layer graph embedding feature, $W^{(l)}$ is a trainable parameter, and $\sigma(\cdot)$ is an activation function, such as ReLU. This propagation mechanism can achieve joint modeling of cross-node behavior patterns while maintaining local structural consistency, thereby obtaining an embedded feature Z_t that contains upstream and downstream dependency information.

To further improve the discriminability and structure preservation ability of embedding, this paper introduces a structural consistency comparison learning objective. Specifically, for any pair of graph embeddings z_i and z_j, if their corresponding container nodes have a call relationship or structural adjacency relationship, they are defined as a "positive sample pair", otherwise, they are "negative sample pairs". Based on this assumption, the comparison loss is constructed as follows:

$$L_{con} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k \in N(i)} \exp(sim(z_i, z_k)/\tau)}$$

Among them, $sim(\cdot,\cdot)$ represents the cosine similarity function, τ is the temperature coefficient, and $N(i)$ is the set of negative samples. This loss can promote the aggregation of container nodes with association relationships in the graph embedding space and reduce the uncertainty of structural expression.

After the multi-layer graph neural encoding is completed, the structural embedding representation $Z_t = \{z_1, z_2, ..., z_N\}$ is finally input into the fusion module

and jointly optimized with the subsequent temporal modeling module. To make the structural embedding both distinguishable and globally consistent, this paper designs the joint optimization objective function as follows:

$$L_{struct} = L_{con} + \lambda \cdot L_{smooth}$$

$$L_{smooth} = \sum_{(i,j)\in\varepsilon_t} \| z_i - z_j \|_2^2$$

Among them, L_{smooth} is the structural smoothing loss, which is used to maintain the continuity of adjacent node embeddings, and λ is the balancing hyperparameter. The introduction of this loss function in the structure-aware modeling stage significantly enhances the model's ability to model container interactions, so that graph embedding can not only characterize individual behavior characteristics but also express high-order dependencies and propagation paths between nodes, providing more robust and expressive structural representation support for subsequent delay prediction tasks.

To achieve end-to-end modeling from structural relationships to dynamic behavior evolution, this study incorporates the generated node embeddings Z_t from the structure-aware graph modeling module as input features to the multi-scale temporal modeling module, enabling the fusion of spatial and temporal information. In this process, the structural embeddings preserve both the topological dependencies and semantic interactions among containers. They also provide fine-grained behavioral cues that are essential for capturing long-term dependencies and multi-scale fluctuations in latency. This design forms the structural foundation for the second innovation, Multi-Scale Temporal Encoding (MSTE), allowing the model to sensitively capture latency dynamics across different temporal granularities. The structure and temporal submodules are connected through internal information flow, forming a structure-driven spatiotemporal modeling mechanism. This provides unified, efficient, and interpretable feature support for subsequent latency prediction tasks.

3.2 Multi-Scale Temporal Encoding

This study introduces a Multi-Scale Temporal Encoding (MSTE) mechanism specifically designed to capture the intricate and dynamic nature of container state transitions over time. The core objective of this module is to model latency-related features at multiple temporal granularities, allowing for a more nuanced understanding of temporal dependencies. By simultaneously focusing on broad, long-term temporal trends and fine-grained, short-term fluctuations, the MSTE module is able to extract rich temporal patterns that are often overlooked in single-scale encoding methods. This dual-level temporal representation enhances the model's sensitivity to both high-frequency disruptions and low-frequency structural shifts, which are critical for recognizing irregular or abnormal behaviors in time series data. The architectural design of this module is illustrated in Figure 3.

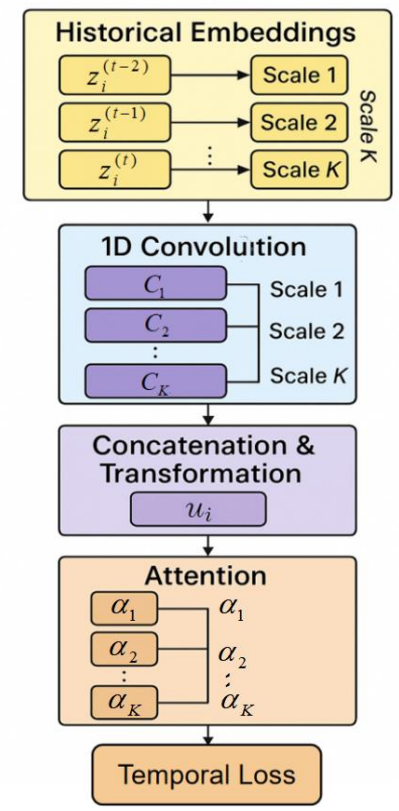

Figure 3. MSTE module architecture

Suppose the node embedding output by the structure modeling module is $z_i \in R^d$, and its evolution in continuous time steps is represented by $\{z_i^{(t-s)},...,z_i^{(t)}\}$. To achieve multi-scale information fusion, this paper introduces multiple sliding window scales $W = \{w_1, w_2,..., w_K\}$ in the time dimension, and each scale corresponds to a local time block. At each time scale, a local convolution operation is performed:

$$c_i^{(w_k)} = Conv1D_{w_k}(z_i^{(t-w_k+1:t)})$$

Among them, $Conv1D_{w_k}(\cdot)$ represents a one-dimensional convolution operation with a window size of w_k, generating a time-aware representation $c_i^{(w_k)}$ of the corresponding scale. Subsequently, all scale features are cascaded and transformed to obtain a multi-scale fusion representation:

$$u_i = \mathrm{Re}LU(W_u \cdot [c_i^{(w_1)} \| c_i^{(w_2)} \| ... \| c_i^{w_K}] + b_u)$$

$\|$ represents the feature concatenation operation, W_u and b_u are trainable parameters, and the output u_i is the temporal enhancement representation of the node i. To capture the significant differences between temporal levels, an attention mechanism is further introduced to weight the features of each scale:

$$a_k = \frac{\exp(a^T \cdot \tanh(W_a \cdot c_i^{(w_k)}))}{\sum_{j=1}^{K} \exp(a^T \cdot \tanh(W_a \cdot c_i^{(w_j)}))}$$

$$v_i = \sum_{k=1}^{K} a_k \cdot c_i^{(w_k)}$$

Among them, a and W_a are attention weight parameters, and v_i is the final time representation after fusion of attention weights. This mechanism significantly improves the model's sensitivity to key time scales and avoids representation offsets caused by redundant window interference.

To strengthen the correlation between time representation and delay prediction goals, this paper designs a loss function based on time residual regression. Specifically, let the actual delay of node i at time $t+1$ be $y_i^{(t+1)}$ and the predicted value be $\hat{y}_i^{(t+1)} = f(v_i)$. Then the optimization goal of the time modeling module is as follows:

$$L_{temp} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i^{(t+1)} - y_i^{(t+1)})^2$$

This loss function not only guides the multi-scale temporal modeling module to generate embeddings that are highly relevant to the prediction targets but also dynamically adjusts the attention weights of different temporal scales during backpropagation. By minimizing L_{temp}, the model can more accurately capture the system's dynamic evolution and develop effective sensitivity to abrupt changes and periodic patterns. This provides strong temporal feature support for container-level network latency prediction.

4. Experimental Results

4.1 Dataset

The base dataset used in this study is "Cleaned Network Latency Estimating Data." This dataset simulates network latency behavior in multi-node distributed systems. It contains multi-dimensional features such as node topology, hop count, round-trip time (RTT), bandwidth, and delay jitter. These characteristics make it highly suitable for container-level network latency modeling.

The dataset includes node-level latency time series and path dependency information. It supports the construction of time series windows for temporal feature modeling and provides adjacency path data for use in graph structure modeling. The latency measurements contain RTT statistics for multiple node pairs over time, along with snapshots of node connection topologies. These are ideal for implementing Structure-Aware Graph Modeling and extracting sliding time windows for Multi-Scale Temporal Encoding.

The entire dataset is stored in CSV format. Key fields include timestamp, source node ID, target node ID, path hop count, RTT mean, RTT variance, and network bandwidth indicators. Researchers can use this information to extract historical latency embeddings for each container node, build communication adjacency matrices, and perform structural feature propagation. The dataset is public, complete, and temporally coherent. It has been used in several network performance prediction tasks and is available on Kaggle for direct search and download. It is well-suited for research on container-level latency prediction.

This dataset does not contain container runtime load data. However, it allows mapping of simulated container instances to the network topology. This makes it possible to transform the original data into communication paths and latency variation sequences between container nodes. Using the raw RTT sequences and topological information, the spatiotemporal hybrid modeling framework can be constructed. On one hand, the structural module uses node-edge information to build dynamic adjacency matrices. On the other hand, the temporal module extracts latency features from time series using multiple window scales. This provides the input for the MSTE module and enables fine-grained network latency prediction.

4.2 Experimental setup

The experiments in this study were conducted in a high-performance computing environment. The system was configured with Ubuntu 22.04 LTS as the operating system. An NVIDIA RTX 3090 GPU with 24 GB of memory was used as the primary acceleration hardware. The system also included an Intel Xeon Gold 6226R CPU with 32 cores at 2.9 GHz and 256 GB of DDR4 memory. This setup ensured sufficient computational and memory resources for handling high-dimensional time series and graph-based modeling tasks.

The experimental framework was built using PyTorch 2.0 and Python 3.10. The graph neural network components were accelerated with DGL (Deep Graph Library) to support efficient structure-aware message propagation. All model training was performed on the GPU. The Adam optimizer was used for parameter updates, with an initial learning rate of 0.001. The batch size was set to 64. A learning rate decay strategy was applied, with a decay rate of 0.95 every 10 epochs.

For hyperparameter settings, the temporal window length was set to 10. The node embedding dimension was 128. The graph neural network had 2 layers. The kernel sizes for the multi-scale convolutional operations were 3, 5, and 7. The dropout rate was set to 0.2. An early stopping strategy was adopted for training. The maximum number of epochs was 200, and the patience on the validation set was set to 10 to prevent overfitting. These configurations were designed to balance model expressiveness and training stability.

4.3 Experimental Results

1) Comparative experimental results

This paper first conducts a comparative experiment, and the experimental results are shown in Table 1.

Table 1: Comparative experimental results

Model	MSE	MAE	MAPE (%)	RMSE
Informer [21]	0.183	0.276	12.4	0.428
Autoformer [22]	0.165	0.253	11.1	0.406
FEDformer [23]	0.142	0.238	10.3	0.376
Crossformer [24]	0.128	0.221	9.70	0.357
Ours (MSTE-SAGM)	0.115	0.209	8.90	0.339

The experimental results show that the proposed model, which integrates structure-aware and multi-scale temporal modeling (MSTE-SAGM), outperforms several representative time series prediction models published in the literature across multiple key evaluation metrics. In particular, it achieves lower error values in MSE and RMSE, indicating stronger overall fitting ability and better global trend capture. Compared with traditional Transformer-based architectures such as Informer and Autoformer, the proposed model demonstrates higher stability and generalization when handling highly dynamic and heterogeneous network latency sequences. This suggests that the incorporation of structural modeling and multi-scale mechanisms contributes positively to capturing latency patterns.

For MAE and MAPE metrics, MSTE-SAGM also shows clear advantages. This indicates that the model achieves better fine-grained error control and provides more accurate estimation of node-level network state fluctuations. Traditional models often suffer from accumulated errors when processing long sequences. This issue becomes more prominent in containerized systems with bursty and non-stationary latency. The multi-scale temporal convolution mechanism introduced in this study effectively mitigates this problem. By enhancing temporal representation through multiple receptive window sizes, the model better captures latency peaks and edge-state trends.

In addition, compared with recent models such as FEDformer and Crossformer that focus on frequency-domain modeling or cross-attention mechanisms, MSTE-SAGM benefits from the Structure-Aware Graph Modeling module. This module introduces communication relationships and service dependency structures among containers. It further enhances the model's ability to capture system-level topological and behavioral interactions. The structure-guided temporal modeling approach introduces semantically meaningful constraints into the prediction process. This improves the model’s ability to respond to abnormal chain propagation and local deviations.

In summary, the proposed model outperforms existing methods across multiple metrics. This advantage is primarily due to the joint use of structure-aware and multi-scale temporal encoding mechanisms. The model demonstrates strong robustness and high interpretability in container-level network latency prediction tasks. It effectively supports fine-grained modeling and early warning of latency behaviors in microservice systems and offers a feasible approach for performance optimization under high concurrency and dynamic conditions.

2) Ablation Experiment Results

To further verify the effectiveness and contribution of each key module in the model, ablation experiments were conducted in this study. By progressively removing or replacing critical components, the ablation study helps assess the role of each module in the overall performance. This validates the rationality and necessity of the model design. Such methodology has been widely used in the structural analysis of complex models and provides clear guidance for future optimization. Table 2 presents the performance changes under different ablation settings across four core evaluation metrics: MSE, MAE, MAPE, and RMSE. By comparing with the full model, the impact and contribution of each submodule to the overall performance can be observed. This table provides direct support for the subsequent analysis of module contributions.

Table 2: Ablation Experiment Results

Method	MSE	MAE	MAPE (%)	RMSE
Baseline	0.192	0.287	13.20	0.438
+SAGM	0.151	0.245	10.70	0.389
+MSTE	0.139	0.234	10.20	0.373
+ All(MSTE-SAGM)	0.115	0.209	8.900	0.339

As shown in the ablation results in Table 2, the introduction of the Structure-Aware Graph Modeling (SAGM) module significantly reduces the overall prediction error. This indicates that communication structures and dependency relationships among nodes have a strong impact on latency variations in container-level network environments. By constructing dynamic graph structures and incorporating structural propagation mechanisms, the model can more effectively capture potential service chain dependencies and upstream-downstream interactions. This enhances the model's ability to represent latency patterns under complex propagation paths. The mechanism also improves the consistency of node representations and provides structural semantic support for subsequent temporal modeling.

When only the Multi-Scale Temporal Encoding (MSTE) module is applied, the model also shows a clear performance improvement. This demonstrates that information from different temporal granularities is complementary for modeling latency trends in dynamic container environments. Through multi-scale convolutional perception paths, the model can detect both local fluctuations and global patterns. This avoids feature bias and information loss caused by using a single time scale. In addition, the scale attention mechanism enables the model to assign dynamic weights to different periods, which improves its responsiveness to sudden latency changes and enhances modeling robustness.

The complete model, which integrates both the structure modeling and the multi-scale temporal modeling modules, further improves performance across all metrics. This confirms the complementary nature of the two components. The

structure module provides explicit prior knowledge of dependencies between nodes, which effectively constrains the spatial distribution of temporal encoding. This allows temporal features extracted at different scales to aggregate with topological consistency. Meanwhile, the temporal module enhances the context-awareness of structural representations through dynamic semantic fusion. Together, they form a bidirectional enhancement from topology-guided to time-driven modeling. This collaborative mechanism strengthens the model's ability to capture dynamic system behavior and provides both theoretical support and practical feasibility for robust and generalizable container-level latency prediction.

3) Effect of contrast loss temperature coefficient on the discriminability of embedding space

This paper also analyzes the impact of the contrast loss temperature coefficient on the discriminability of the embedding space. The experimental results are shown in Figure 5.

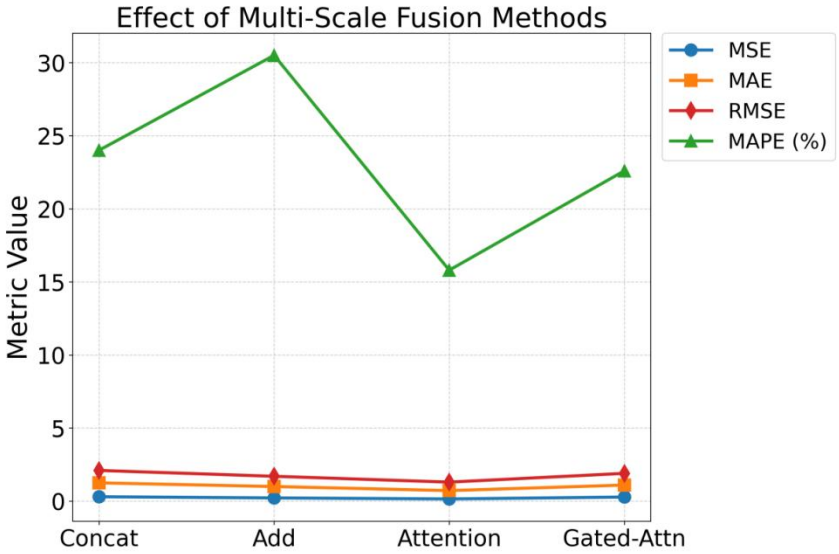

Figure 5. Effect of contrast loss temperature coefficient on the discriminability of embedding space

The comparison results of multi-scale feature fusion methods show that the granularity of feature fusion directly affects the strength of information expression across different temporal resolutions in the latency sequence. When using simple concatenation, the model must learn cross-scale relationships in subsequent layers. As a result, the overall error metrics remain relatively high. However, the MAPE stays at a moderate level, indicating that while concatenation preserves full feature content, it has limited ability to highlight key time segments and detect local extreme fluctuations.

When multi-scale features are merged by element-wise addition, the average errors, such as MAE and RMSE, decrease slightly. However, the MAPE increases significantly. This suggests that linear combination under long-term dependency conditions may cause amplitude drift, which amplifies proportional errors. This phenomenon indicates that in container-level latency prediction, simple addition weakens the complementarity between windows and introduces noise, making the model more sensitive to sudden jitter.

With the introduction of the attention mechanism, all four metrics reach their best values. The improvements in MSE, MAE, and RMSE are particularly notable. This shows that adaptive weight assignment can build effective selection paths between global and local features. It emphasizes the contribution of key scales to prediction outcomes. The significant reduction in MAPE also confirms the attention mechanism's role in suppressing peak deviation. It highlights that different temporal scales have unequal impacts on the evolution of container network latency and require dynamic modeling.

When a gating mechanism is added on top of attention, the model maintains low absolute errors. However, MSE and RMSE rebound, suggesting that excessive gating may create competition between low-frequency and high-frequency information. This weakens the ability to capture long-term dependencies. The results indicate that stronger gating is not always better. It is necessary to balance adaptive weighting and information fidelity. Only by highlighting critical time scales without overly suppressing useful signals can robust modeling of container-level latency sequences be achieved.

4) Analysis of the impact of time window sliding step size on delay fluctuation modeling capabilities

This paper further gives the impact of the time window sliding step on the delay fluctuation modeling capability, and the experimental results are shown in Figure 6.

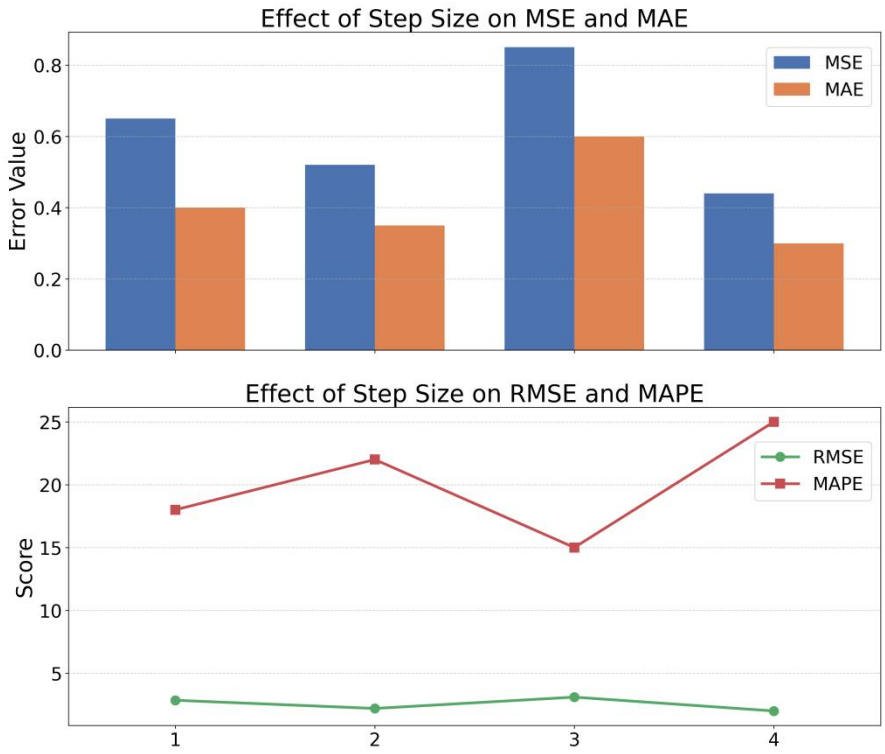

Figure 6. Analysis of the impact of time window sliding step size on delay fluctuation modeling capabilities

In the sensitivity experiments on sliding step size of the time window, we observed that smaller step sizes, such as 1 or 2, lead to better performance in accuracy metrics like MAE and MSE. This suggests that short sliding distances help capture fine-grained dynamic changes in the time series. Such a setting benefits precise modeling of latency fluctuations, especially in complex microservice or containerized environments, where subtle variations often indicate early signs of performance degradation or system anomalies.

However, as the sliding step size increases, the model exhibits fluctuations in RMSE performance. This indicates that a larger step size may weaken the model's ability to capture long-term behaviors, particularly in highly non-stationary network latency sequences. This trend shows that while larger step sizes improve efficiency in processing historical information, they reduce local temporal dependency, which negatively affects overall reconstruction quality.

For the MAPE metric, the model's sensitivity to step size shows a nonlinear pattern. In some configurations, such as a step size of 3, lower relative errors appear. This suggests that a moderate step range may be more adaptive in certain anomalous regions. It also indicates that step size selection affects not only absolute errors but also the ability to model relative changes in latency values, requiring a balance between global trend modeling and local error control.

Considering all evaluation metrics, the effect of step size on latency sequence modeling is multi-dimensional. There is a clear tension between temporal sensitivity and structural awareness. Balancing fine-grained responsiveness with long-term dependency modeling is crucial for improving both the generalization and structural stability of the model.

5. Conclusion

This study addresses the spatiotemporal dynamics of container-level network latency by proposing a prediction framework that integrates structure-aware graph modeling with multi-scale temporal modeling. The Structure-Aware Graph Modeling (SAGM) mechanism is introduced to effectively capture complex interaction dependencies among containers, which enhances the support of structural information for temporal modeling. Based on this, the Multi-Scale Temporal Encoding (MSTE) module is designed to enable parallel modeling and semantic fusion across multiple temporal granularities. This strengthens the model’ s ability to represent non-stationary fluctuation patterns and periodic disturbances.

The proposed joint modeling framework is built around the core idea of heterogeneous information fusion. Structural graph modeling guides the temporal modeling module to identify potential dependency paths and propagation chains, uncovering implicit causal relationships among containers. At the same time, the MSTE module uses convolutional perception and multi-scale encoding to aggregate representations across different temporal levels. This design helps mitigate problems such as local anomaly masking and global trend weakening. The overall architecture demonstrates a deep integration of structure-time co-modeling and provides a unified and robust solution for high-dimensional, multi-source temporal prediction tasks in container-level systems.

In terms of submodule design and modeling strategy, this study further analyzes how multi-scale fusion methods affect the quality of temporal representation. The experiments show that the fusion strategy significantly impacts how features at different scales are aggregated and affects the final prediction performance. A well-designed fusion path enhances the expressiveness of temporal modeling and strengthens the alignment between multi-scale semantics and structural guidance. These results confirm the importance of inter-module collaboration for performance improvement and offer valuable insights for future optimization in complex temporal scenarios.

Additionally, this study systematically evaluates the effect of the time window sliding step on modeling latency fluctuations from the perspective of input mechanisms in temporal modeling. By adjusting sequence sampling intervals and sliding strategies, the model’s responses to sudden jitter, periodic disturbances, and structural delay anomalies are analyzed under different time granularities. The experiments demonstrate that an appropriate time window advancement strategy can effectively balance the trade-off between short-term noise suppression and long-term dependency capture. This improves adaptability to non-stationary latency sequences. These findings expand the flexibility of the proposed method in input modeling and provide both theoretical and practical support for future integration of multi-scale temporal features.

References

[1] Ahmed N, Schmidt-Thieme L. Structure-aware decoupled imputation network for multivariate time series[J]. Data Mining and Knowledge Discovery, 2024, 38(3): 1006-1026.

[2] D. Wu, "Federated Deep Learning with Contrastive Representation for Node State Identification in Distributed Systems," Transactions on Computational and Scientific Methods, vol. 4, no. 8, 2024.

[3] C. Hu, Z. Cheng, D. Wu, Y. Wang, F. Liu and Z. Qiu, "Structural generalization for microservice routing using graph neural networks," Proceedings of the 2025 3rd International Conference on Artificial Intelligence and Automation Control (AIAC), pp. 278-282, 2025.

[4] Xu N, Kosma C, Vazirgiannis M. TimeGNN: temporal dynamic graph learning for time series forecasting[C]//International Conference on Complex Networks and Their Applications. Cham: Springer Nature Switzerland, 2023: 87-99.

[5] Y. Wang, "Adaptive Graph Construction and Spatiotemporal Contrastive Learning for Intelligent Cloud Service Monitoring," 2026.

[6] D. Novaković, N. Vasić, S. Novaković, D. Kostić and R. Bianchini, "{DeepDive}: Transparently Identifying and Managing Performance Interference in Virtualized Environments," Proceedings of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), pp. 219-230, 2013.

[7] Z. Huang, J. Yang, S. Li, C. Zhang, J. Chen and C. Xu, "Shared Representation Learning for High-Dimensional Multi-Task Forecasting under Resource Contention in Cloud-Native Backends," arXiv preprint arXiv:2512.21102, 2025.

[8] Naghashi V, Boukadoum M, Diallo A B. A multiscale model for multivariate time series forecasting[J]. Scientific Reports, 2025, 15(1): 1565.

[9] J. Yang, J. Chen, Z. Huang, C. Xu, C. Zhang and S. Li, "Cost-TrustFL: Cost-Aware Hierarchical Federated Learning with Lightweight Reputation Evaluation across Multi-Cloud," arXiv preprint arXiv:2512.20218, 2025.

[10] Karadag Y M, Kalkan S, Dino I G. ms-Mamba: Multi-scale Mamba for Time-Series Forecasting[J]. arXiv preprint arXiv:2504.07654, 2025.

[11] B. Chen, "FlashServe: Cost-Efficient Serverless Inference Scheduling for Large Language Models via Tiered Memory Management and Predictive Autoscaling," 2025.

[12] Y. Xing, Y. Deng, H. Liu, M. Wang, Y. Zi and X. Sun, "Contrastive Learning-Based Dependency Modeling for Anomaly Detection in Cloud Services," arXiv preprint arXiv:2510.13368, 2025.

[13] Cheng S, Liu Q, Jin H, et al. Collaborative optimization of truck scheduling in container terminals using graph theory and DDQN[J]. Scientific Reports, 2025, 15(1): 6950.

[14] Z. Zhang, W. Liu, J. Tao, H. Zhu, S. Li and Y. Xiao, "Unsupervised Anomaly Detection in Cloud-Native Microservices via Cross-Service Temporal Contrastive Learning," 2025.

[15] Q. Zhang, N. Lyu, L. Liu, Y. Wang, Z. Cheng and C. Hua, "Graph Neural AI with Temporal Dynamics for Comprehensive Anomaly Detection in Microservices," arXiv preprint arXiv:2511.03285, 2025.

[16] H. Liu, Y. Kang and Y. Liu, "Privacy-Preserving and Communication-Efficient Federated Learning for Cloud-Scale Distributed Intelligence," 2025.

[17] Cao D, Wang Y, Duan J, et al. Spectral temporal graph neural network for multivariate time-series forecasting[J]. Advances in neural information processing systems, 2020, 33: 17766-17778.

[18] Cini A, Zambon D, Alippi C. Sparse graph learning from spatiotemporal time series[J]. Journal of Machine Learning Research, 2023, 24(242): 1-36.

[19] Z. L. Li, G. W. Zhang, J. Yu and L. Y. Xu, "Dynamic graph structure learning for multivariate time series forecasting," Pattern Recognition, vol. 138, p. 109423, 2023.

[20] Y. Wang, H. Liu, G. Yao, N. Long and Y. Kang, "Topology-aware graph reinforcement learning for dynamic routing in cloud networks," arXiv preprint arXiv:2509.04973, 2025.

[21] Zhou H, Zhang S, Peng J, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(12): 11106-11115.

[22] Wu H, Xu J, Wang J, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting[J]. Advances in neural information processing systems, 2021, 34: 22419-22430.

[23] Zhou T, Ma Z, Wen Q, et al. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting[C]//International conference on machine learning. PMLR, 2022: 27268-27286.

[24] C. Wang, T. Yuan, C. Hua, L. Chang, X. Yang and Z. Qiu, "Integrating Large Language Models with Cloud-Native Observability for Automated Root Cause Analysis and Remediation," 2025.