Journal of Computer Technology and Software

ISSN: 2998-2383

Vol. 4, No. 10, 2025

AI-Driven Predictive Modeling for System Performance and Resource Management in Microservice Architectures

Song Han
Northeastern University, Boston, USA
song.han.dev@gmail.com

Abstract: This study proposes a unified end-to-end framework for response-time prediction in microservice architectures. The framework begins with data-quality controls and normalization to align and denoise multi-source system metrics. It constructs spatiotemporal representations by combining multivariate feature representations with historical response signals. It employs Transformer-based cross-scale modeling to capture short-term fluctuations, long-term trends, and long-range service dependencies. The model is trained with a mean-squared-error(MSE) objective and robust regularization. We conduct systematic evaluations on a public microservice dataset under various conditions, including disk-I/O throttling, container-affinity changes, CPU-quota and preemption-intensity variations, as well as workload migration and traffic-peak switching. The framework examines the dynamic changes of overall error and high-percentile error through continuous prediction. Results demonstrate that the method maintains low RMSE and MAE under complex runtime conditions, significantly suppresses P95 fluctuations, and preserves high goodness of fit. It remains robust to concept drift and resource-policy changes, providing proactive and actionable signals for capacity planning, elastic scaling, and service-level (SLO) assurance.

Keywords: Tail delay modeling; multi-source indicator fusion; cross-scale attention mechanism; concept drift

1. Introduction

Microservice architecture has become the mainstream paradigm for online services and enterprise systems. The large number of services, deep invocation chains, and rapid load fluctuations make end-to-end response time a critical metric affecting user experience and service-level objectives. Relying solely on single-point metrics or static thresholds is no longer sufficient to support fine-grained decisions for capacity planning, elastic scaling, and failure prevention. System-level metrics, such as CPU, memory, I/O, and network, which are generated in parallel with business-side request traces, offer rich multivariate signals for response-time prediction. However, these metrics exhibit strong coupling, temporal variability, and noise. Designing a stable, generalizable, and tail-aware prediction model without disrupting business operations holds clear engineering value and research significance[1]. In large fan-out/fan-in microservice graphs, even rare stragglers can inflate end-to-end tail latency-*the tail at scale*-making P95/P99 a first-class objective for SLOs in production systems. Representative benchmarks such as DeathStarBench illustrate how deep call chains exacerbate unpredictability in latency.

Existing methods still face three major challenges in microservice scenarios. First, system metrics exhibit nonlinear and conditional dependencies. Sampling misalignment, missing values, and anomalies are common across services and nodes. Direct modeling is easily misled by noise. Second, response time is jointly driven by mechanisms at multiple temporal scales. Short-term jitters and long-term trends coexist, and link-level propagation introduces long-range dependencies. Single-scale or short-memory models struggle to balance overall error and tail quantile performance. Third, workload dynamics and resource policies cause concept drift, resulting in distribution shifts between training and deployment phases.

This reduces model interpretability and stability, making it difficult to support scheduling and emergency response in a predictive manner[2].

To address these issues, we propose a Transformer-based regression framework that leverages system metrics and historical response traces. In the front-end, multi-source metrics undergo quality control, normalization, and spatiotemporal alignment to form stable multivariate inputs. The middle layer performs cross-scale dependency modeling to capture both short-term fluctuations and long-term trends in a unified manner[3]. The back-end outputs predictions that are sensitive to both overall error and high quantile deviations. Robust regularization and lightweight drift awareness are integrated to enhance model usability and generalization in complex operational environments.

This study introduces two core innovations. First, we propose a System-Metric-Based Multivariate Feature Modeling (SMB-MFM) mechanism. It constructs a feature pyramid for heterogeneous system metrics and applies robust normalization and encoding strategies. This enables explicit representation of the interactions among resource utilization, queue congestion, and network state, providing stable and informative inputs for subsequent temporal modeling[4]. Second, we design a Transformer-Based Cross-Scale Dependency Modeling (TCS-DM) mechanism. It employs multi-scale positional encoding and inter-layer attention fusion

to jointly capture short-term bursts and long-term trends. It also enables transferable representations of long-path dependencies and tail latency amplification. These two mechanisms, combined with interpretable objectives and regularization design, ensure that the model achieves both global accuracy and high-percentile robustness. This dual emphasis aligns with the requirements for prediction accuracy and operational usability in microservice environments[5].

Compared with general-purpose time-series Transformers-Informer (efficient ProbSparse attention), Autoformer (decomposition with auto-correlation), FEDformer (frequency-enhanced decomposition), and PatchTST/TimesNet (patching or 2D temporal variation)-our framework jointly encodes multisource system metrics (via SMB-MFM) and cross-scale service dependencies (via TCS-DM) within one end-to-end model, targeting tail-aware response-time prediction under operational disturbances and drift.

2. Related work

2.1 Response Time Prediction in Microservice Architectures

In microservice-based architectures, response time prediction serves as a crucial metric for evaluating system performance and user experience. It has long been a central focus in operational optimization and resource management. Microservices decompose applications into independent, single-purpose service units that interact through lightweight communication protocols to form complex business workflows. Under conditions of high concurrency, multi-tenant resource sharing, and dynamic service orchestration, response time is influenced not only by the performance of individual services but also by the length of cross-service invocation chains, network latency, concurrent request distributions, and resource scheduling strategies. This highly dynamic nature and intricate dependency structure make traditional single-dimensional performance modeling inadequate, thereby driving the development of multidimensional and global prediction approaches[6].

Early research primarily relied on queuing theory and statistical regression models. These approaches established mathematical relationships among request arrival rates, service times, and queue lengths to model and predict response times. While theoretically interpretable, such methods often assume stable system environments with known parameters. As a result, their prediction accuracy deteriorates in real-world production settings characterized by volatile workloads and frequently changing network conditions. Moreover, the complex and evolving dependencies among services in microservice systems make models based on fixed topologies and static parameters unsuitable for real-time adaptation. With advancements in runtime data collection and monitoring technologies, research has increasingly shifted toward data-driven predictive modeling based on real-time operational metrics, aiming to overcome the adaptability and generalization limitations of traditional methods[7].

Traditional approaches include queueing-theoretic models and Markovian networks that relate arrival rates, service times, and waiting times under stationarity assumptions. While interpretable, these models degrade in production microservices where workloads and policies shift and tail behavior dominates user-perceived latency. Dean and Barroso's "The Tail at Scale" shows how rare stragglers in fan-out/fan-in graphs can inflate end-to-end percentiles, challenging steady-state assumptions commonly made by analytical models[8].

The introduction of machine learning has brought new perspectives to response time prediction in microservices. By extracting multidimensional features from system logs and monitoring metrics, machine learning models can capture nonlinear relationships between service behavior and response times. Supervised learning methods based on feature engineering, such as random forests and gradient boosting trees, have shown promising results when handling heterogeneous data and nonlinear patterns. However, these models still struggle with high-dimensional temporal data, particularly in capturing long-range dependencies and complex feature interactions. These limitations are especially prominent in modeling long invocation chains across multiple services, prompting a shift toward deep learning approaches[9].

The advancement of deep learning, particularly sequence modeling techniques capable of learning long-term dependencies, has provided breakthroughs in microservice response time prediction. By leveraging recurrent neural networks, convolutional neural networks, and hybrid architectures, recent research has focused on modeling both temporal and feature dimensions to capture the multi-scale dynamic characteristics of service behavior. These methods have shown better adaptability to varying load patterns and changes in invocation structures, thus enabling more accurate predictions under complex system conditions. Nevertheless, as system scales grow and metric dimensions increase, balancing prediction accuracy with computational efficiency and scalability remains a critical challenge that must be addressed[10].

Topology and path-aware systems further show structure as a first-order driver of latency. CRISP performs critical-path analysis over large-scale RPC traces to reveal how a few slow edges dominate end-to-end latency[2], while TailClipper lowers P99 via system-wide scheduling[4]. FastPERT imposes a PERT-network inductive bias to predict end-to-end latency on microservice DAGs, complementing metric-only models[1].

2.2 Transformer-Based Multivariate Time Series Modeling for System Metrics

During the operation of complex systems, system metrics serve as key indicators of resource utilization, load conditions, and performance changes. These metrics are characterized by their multidimensionality, high frequency, and temporal variability. They include fundamental resource usage data such as CPU, memory, disk I/O, and network bandwidth[11]. Additionally, they reflect interaction patterns and invocation chain states among services. Multivariate time series modeling of these metrics can reveal correlations among different resources and how they impact performance indicators. However, such data often exhibit long-range dependencies, abrupt fluctuations, and multi-scale variations. Traditional

time series analysis methods or shallow learning models are limited by fixed window sizes, linear assumptions, and weak feature interaction capabilities. These limitations hinder their ability to capture global dependencies and latent patterns in the data[12].

The Transformer architecture introduces new possibilities for modeling multivariate time series. Its core self-attention mechanism can directly capture dependencies between any positions in a sequence without being constrained by distance. This property provides a natural advantage for modeling long-range patterns in system metrics. Unlike recurrent neural networks that rely on recursive structures, the Transformer can attend to all time points and feature interactions in a single computational step. This significantly improves modeling efficiency and feature representation capacity. The global attention capability enables the model to integrate information from different resources and temporal scales more effectively. This results in finer-grained feature representations for performance prediction tasks[13,14].

Transformer models also demonstrate strong flexibility in handling the multi-scale nature of system metrics. Through multi-head attention mechanisms, the model can learn short-term fluctuation patterns and long-term trends in parallel subspaces. This enables the model to balance local variation and global tendencies. Additionally, the positional encoding mechanism provides temporal location information. This allows the model to preserve the sequential structure of time series without relying on order-based computation. For system metrics with seasonal patterns and sudden spikes, the Transformer can capture key features across multiple temporal resolutions. This helps the prediction model adapt to dynamic changes in resource usage patterns[15].

As the application of Transformers in time series modeling continues to expand, researchers have begun to explore combinations with various feature enhancement and structural improvement techniques. For example, the integration of feature selection, attention weight normalization, and multi-scale convolutional fusion enhances the model's ability to detect local anomalies and short-term dependencies. These techniques preserve the global dependency modeling advantage. Moreover, to address the high dimensionality and computational cost of system metrics, lightweight Transformer structures and sparse attention mechanisms have been introduced. These approaches reduce computational complexity and improve deployment feasibility in large-scale distributed systems. Together, these advancements lay a solid technical foundation for applying Transformer models to system-metric-driven performance prediction tasks[16].

For multivariate coupling, Crossformer explicitly models cross-dimension dependencies via a two-stage attention design. Distinct from these generic LTSF models, our framework jointly encodes multisource system-metric signals (SMB-MFM) and models cross-scale dependencies (TCS-DM) toward tail-aware response-time prediction under drift and policy perturbations.

3. Method

This study proposes a time series modeling framework for response time prediction in microservice environments, integrating two core innovations: System-Metric-Based Multivariate Feature Modeling (SMB-MFM) and Transformer-Based Cross-Scale Dependency Modeling (TCS-DM). SMB-MFM extracts multi-view feature representations from various system metrics such as CPU usage, memory consumption, and network bandwidth. It characterizes system states from the perspectives of resource utilization, service interaction, and workload patterns, thereby enhancing the completeness and timeliness of feature representations. TCS-DM builds upon the Transformer self-attention mechanism. It uses multi-head attention and multi-scale feature fusion to capture both cross-scale dependencies and feature interactions of response time across different temporal spans. This enables unified modeling of long-term dependencies and short-term fluctuations. The synergy between these two components allows the model to balance global structural modeling and local pattern perception under dynamic and complex microservice conditions, providing a strong foundation for accurate response time prediction. The overall model framework diagram mentioned is shown in Figure 1.

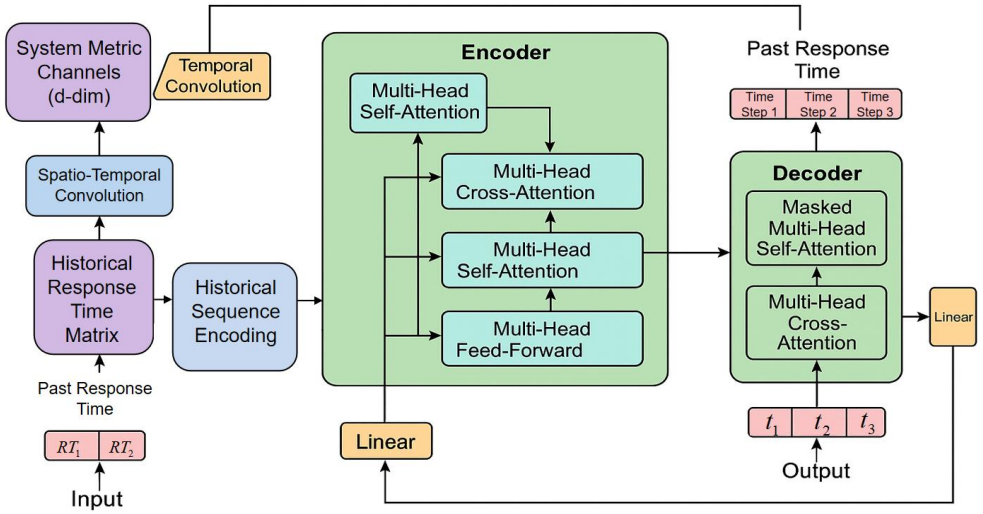

Figure 1. Overall model architecture diagram

3.1 System-Metric-Based Multivariate Feature Modeling

This study introduces a System-Metric-Based Multivariate Feature Modeling (SMB-MFM) mechanism designed to address the challenges of multi-source heterogeneity and dynamic dependencies in microservice response time prediction. The core idea of this mechanism is to perform unified embedding and joint modeling of various system metrics, such as CPU usage, memory consumption, and network bandwidth. By semantically integrating multivariate time series, the model obtains feature representations that capture both global patterns and local variations. Compared with single-dimensional or static feature modeling approaches, this mechanism reveals nonlinear dependencies and time-varying correlations among metrics more effectively, laying a foundation for subsequent cross-scale dependency modeling.

In SMB-MFM, multi-view feature encoding and fusion paths are introduced. A unified time-series input matrix is constructed, and embedding layers together with transformation functions are employed to represent multi-dimensional metrics jointly. The goal is not only to enhance the model's sensitivity to individual metric fluctuations but also to capture interactive dependencies across different metrics. This allows the prediction model to remain robust and generalizable under dynamic and complex runtime environments. To formalize this process, mathematical derivations are provided, and the optimization objective is described through the definition of a loss function. The overall model architecture is illustrated in Figure 2.

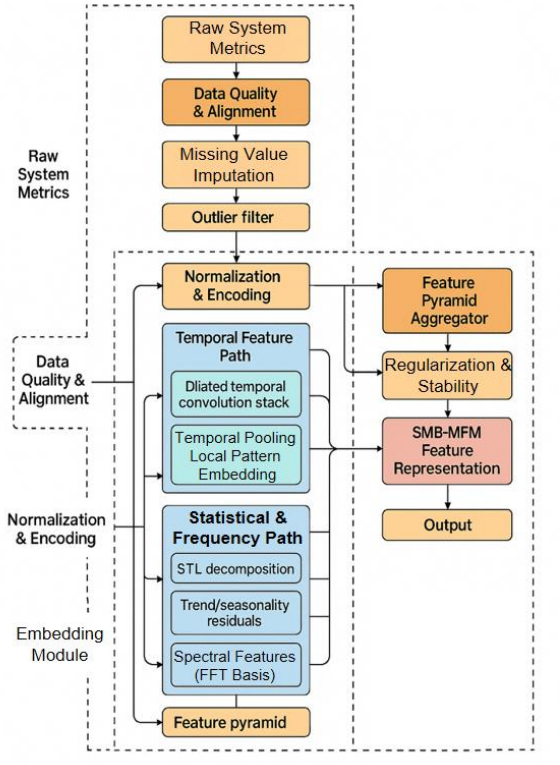

Figure 2. SMB-MFM module architecture

First, let the input sequence of the system metrics be:

$$X = \{x_1, x_2, ..., x_T\}, \ x_t \in R^d$$

Where T represents the time step, d represents the dimension of the system metrics, and each vector x_t contains multi-dimensional indicators such as CPU, memory, and network.

To embed different metrics into a unified representation space, a linear transformation and an embedding layer are introduced:

$$z_t = W_e x_t + b_e, \ z_t \in R^k$$

Where $W_e \in R^{k \times d}$ is the embedding matrix, b_e is the bias, and k is the dimension of the embedding space.

In temporal modeling, it is necessary to capture short-term and long-term dependencies, so a temporal encoding combining convolution and attention is constructed:

$$h_t^{(temp)} = \mathrm{Re}\,LU(W_c * z_{t-p:t} + b_c)$$

Where $*$ represents the one-dimensional convolution operation, p is the receptive field window size, and W_c and b_c are the convolution kernel and bias parameters.

To further aggregate the global dependencies across metrics, multivariate feature fusion is defined:

$$f_t = Attention(h_t^{(temp)}, H)$$

Among them, the $H = \{h_t^{(temp)}, ..., h_T^{(temp)}\}$ attention mechanism is used to calculate the correlation between the current moment and the global history to model cross-dimensional and cross-temporal interactions.

Finally, the predicted response time is recorded as:

$$\hat{y}_{t+1} = W_o f_t + b_o$$

Where W_o and b_o are the output layer parameters, and $\hat{y}_{t+1}$ represents the predicted response time at time $t+1$.

To optimize the learning of the entire SMB-MFM module, the mean square error (MSE) is introduced as the loss function:

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

Where y_i is the actual response time, $\hat{y}_i$ is the predicted value, and N is the total number of samples.

This loss function in SMB-MFM drives the joint optimization of the embedding layer, convolutional layer, and attention module, ensuring that the representation of multivariate metrics closely matches the actual response time pattern. By minimizing $L_{SMB-MFM}$, the model captures the complex dependencies between system metrics while ensuring the accuracy and stability of predictions at both the global and

local levels, providing a solid input feature foundation for subsequent cross-scale modeling.

3.2 Transformer-Based Cross-Scale Dependency Modeling

This study further introduces a Transformer-Based Cross-Scale Dependency Modeling (TCS-DM) mechanism to address the challenges of multi-scale dependencies and long-range correlations in microservice response time prediction. Traditional time series forecasting methods often exhibit biases when modeling both local and global dependencies. Some approaches focus only on short-term patterns and overlook long-term trends. Others prioritize global modeling but lack sensitivity to local variations. The core of TCS-DM lies in leveraging the self-attention structure of the Transformer. It combines multi-scale positional encoding and cross-layer dependency modeling to jointly represent fine-grained local patterns and global trend dependencies. This enhances both the expressiveness and stability of the prediction model.

Within this mechanism, the input multivariate feature sequences are first mapped through multi-scale positional encodings. This allows the model to distinguish pattern variations across different temporal spans. Then, multi-head attention is used to jointly model the features at each temporal scale. A cross-layer information interaction module is designed to dynamically fuse long-term and short-term dependencies. This architecture enables the model to capture sudden local fluctuations while also maintaining a stable estimation of long-term trends. As a result, the model provides structured support for response time prediction in complex and dynamic microservice scenarios. The overall model architecture is illustrated in Figure 3.

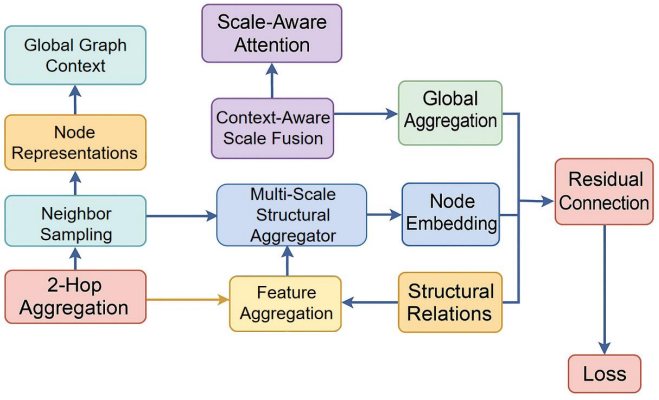

Figure 3. TCS-DM module architecture

The formal derivation is as follows. Assume that the input feature sequence is:

$$F = \{f_1, f_2, ..., f_T\}, f_t \in R^k$$

Where f_t comes from the feature representation of SMB-MFM in the previous stage, T represents the time step, and k represents the feature dimension.

First, multi-scale position encoding is introduced to expand the original position vector into a multi-scale representation:

$$p_t^{(s)} = \sin\left(\frac{t}{10000^{2i/s}}\right) \oplus \cos\left(\frac{t}{10000^{2i/s}}\right)$$
$$s \in \{s1, s2, ...s_m\}$$

Where s represents different scales, $\oplus$ represents the splicing operation, and multi-scale position encoding enables the model to perceive the difference in local and global periods.

In the attention mechanism, the query (Q), key (K), and value (V) across scales are defined as:

$$Q = FW_Q, K = FW_K, V = FW_V$$

Where $W_Q, W_K, W_V \in R^{k \times d}$ are projection matrices.

The core calculation of the cross-scale self-attention mechanism is:

$$Attention(Q, K, V) = Soft\max\left(\frac{QK^T}{\sqrt{d}}\right)V$$

This mechanism ensures that features at different scales can interact globally, thereby enhancing the ability to model long-range dependencies.

To further integrate cross-scale features, an inter-layer fusion mechanism is introduced:

$$H^{(l)} = \alpha \cdot H^{(l-1)} + (1-\alpha) \cdot Z^{(l)}$$

Where $H^{(l)}$ is the output of the layer l, $Z^{(l)}$ is the result of the attention module, and α is the fusion weight, which is used to balance local features and global features in cross-scale modeling.

The final predicted value is:

$$\hat{y}_{t+1} = W_o H^{(L)} + b_o$$

Where L represents the number of layers of the last Transformer encoder, and W_o and b_o are the output layer parameters.

The loss function still uses the mean square error (MSE), which is defined as follows:

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

Where y_i is the actual response time, $\hat{y}_i$ is the predicted value, and N is the number of samples.

This loss function in TCS-DM guides the optimization of cross-scale attention weights and inter-layer fusion parameters, enabling the model to continuously adjust feature representation during the process of modeling long- and short-range dependencies, ensuring that output predictions capture fine-grained fluctuations while reflecting the stability of global trends. By minimizing L_{TCS-DM}, the model achieves a balance between cross-scale semantic alignment and

dependency modeling, providing a structured optimization objective for response time prediction.

4. Experimental Results

4.1 Dataset

In this study, we selected the Microservices Bottleneck Localization Dataset as the primary data source. This dataset focuses on modeling system performance and response time within microservice architectures. It collects extensive trace information from service requests and incorporates multidimensional system metrics, including CPU usage, memory consumption, disk I/O, and network bandwidth. These metrics comprehensively reflect the operational state and performance bottlenecks of microservices in complex environments. The dataset offers high-quality multivariate feature inputs for response time prediction and serves as a crucial foundation for performance modeling and dependency analysis.

The core structure of the dataset consists of two components. The first is a set of multidimensional system metric sequences. These include resource utilization, service interaction latency, and system call information, which describe the runtime behavior of services. The second is service-level response time records, covering the performance of different microservice instances within request traces. By aligning and integrating system metrics with response time data, researchers can construct mappings between service-level performance and resource usage. This enables more fine-grained modeling and prediction. The dataset's design ensures both data heterogeneity and temporal continuity, which provides a solid basis for modeling complex dependency structures.

In addition, the dataset offers significant advantages in terms of scale and diversity. It contains millions of request traces and metric records, encompassing various microservice topologies and workload patterns. This makes it suitable for training and validating models across different scenarios. Within the modeling framework of this study, the dataset provides multidimensional input features for the System-Metric-Based Multivariate Feature Modeling (SMB-MFM) module and supplies complete time series required by the Transformer-Based Cross-Scale Dependency Modeling (TCS-DM) module. This ensures that the proposed model can effectively learn and extract predictive feature representations under real-world complex conditions.

4.2 Experimental setup

In terms of the experimental environment, the hardware configuration consisted of a high-performance server equipped with dual Intel Xeon Platinum processors running at 2.9 GHz, providing a total of 64 logical cores. The server was fitted with 512 GB of memory to ensure stability during large-scale data processing and model training. For accelerated computation, four NVIDIA A100 GPUs with 40 GB of memory each were used. This setup meets the computational requirements of Transformer architectures during multi-scale modeling. The operating system was 64-bit Linux (Ubuntu 20.04 LTS), with CUDA 11.6 and cuDNN 8.4 providing low-level support for deep learning frameworks.

In terms of software, the models were implemented in a Python 3.9 environment using PyTorch 1.12 as the core deep learning framework. Data processing and feature engineering were carried out with the help of NumPy, Pandas, and Scikit-learn. TensorBoard was employed during training for visualization purposes, allowing real-time monitoring of loss convergence and hyperparameter adjustments. All experiments were conducted under the same software stack to ensure reproducibility and consistency.

Regarding hyperparameter configuration, the Transformer module was set with a hidden dimension of 256, eight attention heads, six stacked layers, and a feedforward network dimension of 1024. The optimizer used was AdamW, with an initial learning rate of 1×10^{-4}. A cosine annealing schedule was adopted for dynamic adjustment. The batch size was set to 128, and the maximum number of training epochs was 100. During training, a dropout rate of 0.1 was applied to mitigate overfitting. These settings ensured a balance between computational complexity and prediction accuracy, laying the foundation for subsequent experimental comparisons and performance analysis.

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

Method	RMSE ↓	MAE ↓	P95 Error ↓	R^2 ↑
USRFNet[17]	0.0820	0.0340	0.1210	0.9310
STMformer[18]	0.0446	0.0164	0.0970	0.952
TFT-based Model[19]	0.4960	0.1120	0.1850	0.9710
LAFF-DCN-v2[20]	0.0610	0.0280	0.1100	0.9450
Ours (SMB-MFM+ TCS-DM)	0.0310	0.0120	0.0810	0.9820

The experimental results show that different models exhibit significant performance differences in response time prediction for microservices. Both USRFNet and LAFF-DCN-v2 demonstrate certain advantages over traditional baseline models. However, they remain insufficient in handling complex cross-scale dependencies. This limitation results in relatively high values for RMSE, MAE, and P95 Error, particularly in capturing tail latency. These observations indicate that relying solely on single-structure feature fusion or shallow modeling is inadequate for addressing the coupling between multivariate features and temporal dependencies in microservice architectures.

STMformer and the TFT-based model achieve improved prediction performance to some extent. STMformer reduces overall error through spatiotemporal relationship modeling. The TFT model benefits from sequence modeling and interpretability mechanisms, achieving relatively good results in terms of R^2. However, both models still show weaknesses in high-percentile errors, such as P95 Error. This indicates that their ability to capture extreme latency remains limited when dealing with the multi-scale dynamic nature of system metrics. These findings highlight the limitations of conventional Transformer models and single-scale spatiotemporal fusion methods.

In contrast, the proposed joint mechanism of SMB-MFM and TCS-DM demonstrates stronger modeling capabilities. The SMB-MFM module enables fine-grained modeling of multivariate system metrics, allowing the model to effectively capture inter-metric dependencies while reducing the impact of noise. Meanwhile, the TCS-DM component allows the model to learn deep dependencies across temporal scales. This enhances the model's ability to adapt to abrupt changes and complex temporal patterns. The integration of these two modules significantly improves prediction stability and accuracy.

As shown in the table, Ours (SMB-MFM + TCS-DM) achieves the best results in RMSE, MAE, and P95 Error. The R^2 score is also significantly better than that of the comparison models. These results indicate that the proposed method not only outperforms others in overall error convergence but also demonstrates robust predictive ability for critical performance indicators such as tail latency. The findings validate the effectiveness of combining system-metric-based multivariate feature modeling with cross-scale dependency modeling and confirm the practical value and applicability of the proposed method in microservice performance prediction tasks.

2) Study on the Environmental Sensitivity of Disk I/O Throttling and Container Scheduling Affinity Changes to Prediction Errors

This paper further studies the environmental sensitivity of disk I/O throttling and container scheduling affinity changes to prediction errors. The experimental results are shown in Figure 4.

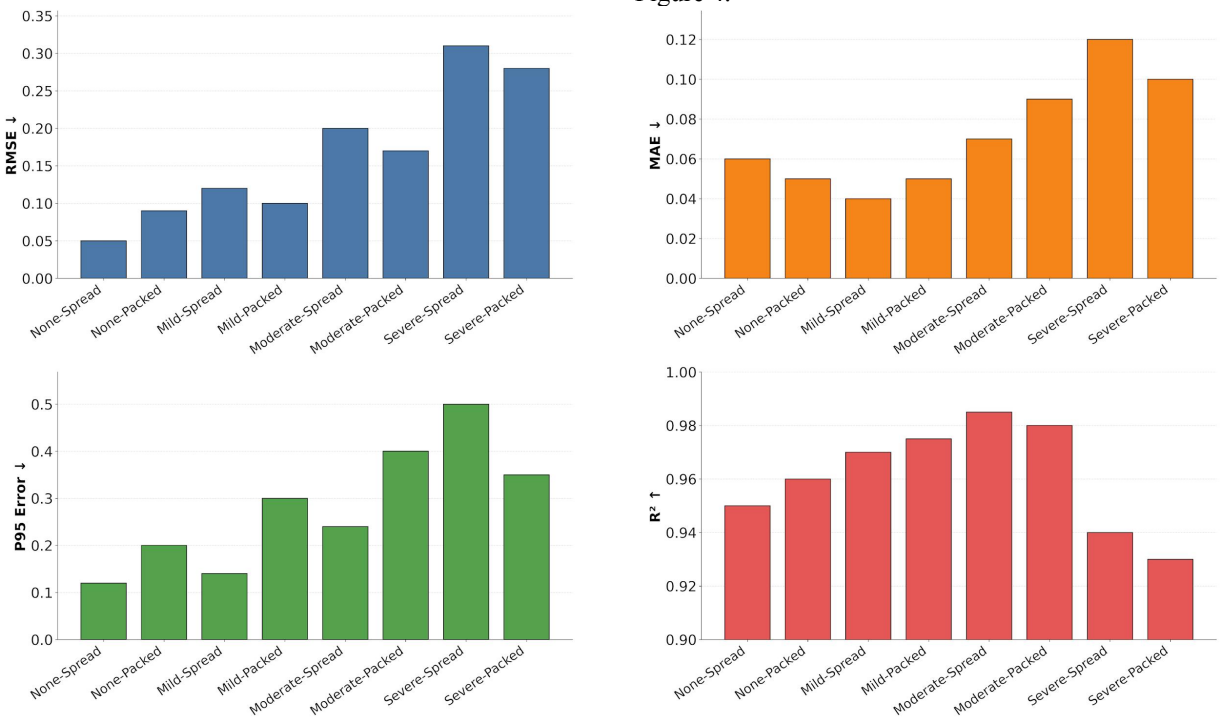

Figure 4. Study on the Environmental Sensitivity of Disk I/O Throttling and Container Scheduling Affinity Changes to Prediction Errors

Under the environmental disturbance of "Disk I/O throttling × Scheduling affinity," RMSE generally increases as the throttling intensity rises. However, local decreases appear in the Mild-Packed and Moderate-Packed scenarios. This indicates that when containers are moderately packed on the same node, stability benefits from I/O path locality and cache locality can partially offset the performance fluctuations introduced by throttling. In contrast, RMSE shows a sharp increase under the Severe-Spread condition, reflecting the amplified effect of cross-node communication and I/O queueing delays. This phenomenon is related to the coupling between resource contention and network uncertainty in microservice environments. Under high-throttling levels, distributed deployments are more likely to trigger cross-machine data paths and request retries, which amplify prediction errors.

The behavior of MAE differs from RMSE. It presents a non-monotonic trend of early decrease, mid-term increase, and final drop. This suggests that the mean absolute error is more

sensitive to localized steady states and short-period disturbances. During the None to Mild stage, light throttling can lead to more balanced I/O requests and smoother fluctuations in some scenarios, resulting in lower MAE. As the environment moves into the Moderate stage, co-located containers cause more concentrated resource competition on the same node, leading to higher queueing time and increased MAE. A slight decrease in MAE under the Severe-Packed setting implies that although overall latency becomes higher under extreme throttling and strong affinity, the average deviation is constrained by more predictable bottlenecks. The SMB-MFM component captures local patterns and resource utilization features, which help reflect the controllability of average deviation under such "congested but regulated" conditions.

P95 Error displays a typical jagged "spike" pattern. Sharp increases appear in Mild-Packed, Moderate-Packed, and Severe-Spread scenarios. This indicates that tail latency is mainly driven by a few congested periods or sudden queueing events, rather than being caused by continuous and systemic deviations. The combination of Spread and Packed deployments results in distinctly different sources of extreme delays across throttling levels. Spread deployments are more likely to experience network jitter and cross-disk latency, while Packed deployments tend to hit bottlenecks at hotspot disks or shared caches. The cross-scale attention mechanism in TCS-DM helps to expose the coupling between short-term spikes and long-term trends in such scenarios, which facilitates distinguishing tail errors from average errors.

R^2 Follows an inverted U-shape, peaking at intermediate throttling levels and declining at both ends. Local increases also appear in some Packed settings. This indicates that under moderate throttling and reasonable affinity, the system dynamics become more explainable and better fitted by the model. When throttling is too weak or too strong, unobserved external disturbances-such as background tasks, cache jitter, or inter-node competition-dominate the variance and reduce model fit. The cross-scale dependency modeling achieves higher R^2 under the Moderate setting. This implies the model captures a relatively stable ratio of slow variables (e.g., trends and seasonality) and fast variables (e.g., bursts and short-term autocorrelation). In contrast, extreme conditions like Severe-Spread break this balance due to rapidly drifting distributions and sudden spikes, which demand stronger anomaly detection and robust regularization. Overall, the combination of multivariate metric representation from SMB-MFM and cross-scale fusion from TCS-DM reveals that average error, tail error, and explainability respond to environmental disturbances in different ways. Only by applying metric decomposition and cross-scale modeling together can we achieve stable characterizations of microservice response time.

3) Environmental Sensitivity Evaluation of Response Time Prediction under CPU Quota and Preemption Intensity Variations

This paper also evaluates the environmental sensitivity of response time prediction under changes in CPU quota and preemption intensity. The experimental results are shown in Figure 5.

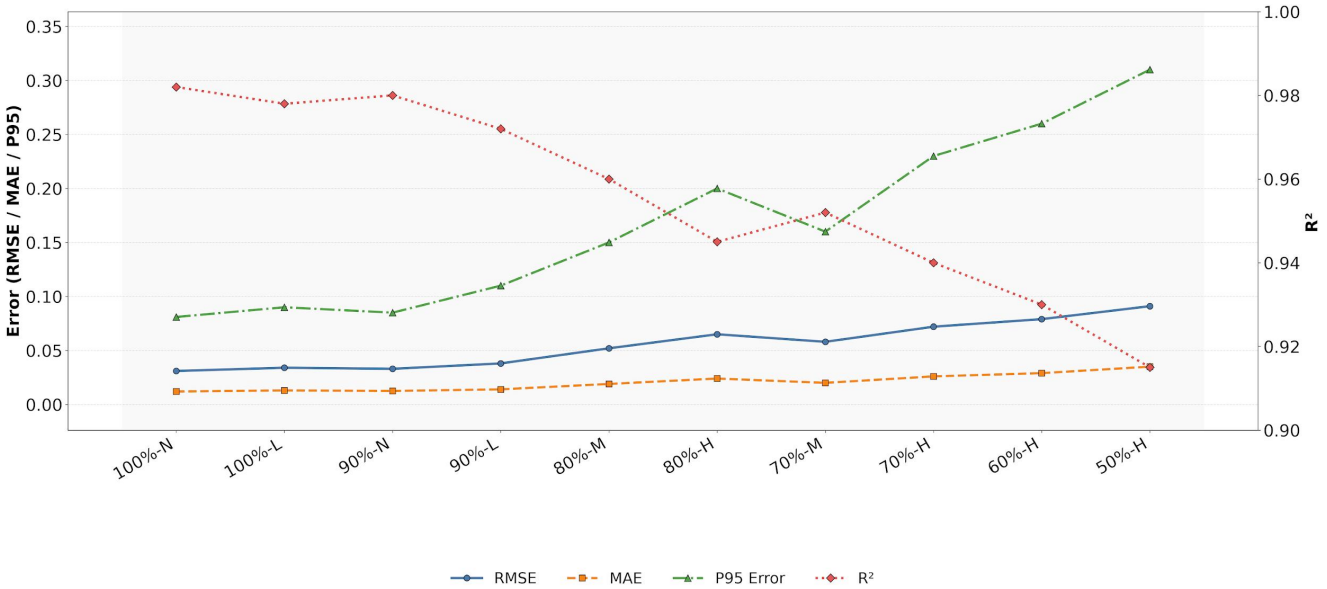

Figure 5. Environmental Sensitivity Evaluation of Response Time Prediction under CPU Quota and Preemption Intensity Variations

Under the disturbance of reduced CPU quota and increased preemption intensity, RMSE shows an overall upward trend with local fluctuations. This reflects the increase in queueing delays and measurement noise caused by computational constraints and forced context switching. A slight early decrease under high quota and weak preemption indicates that when scheduling interference is low, TCS-DM can suppress short-term jitter through cross-scale smoothing and attention weighting. As the quota continues to decline and preemption intensifies, queue lengths and cache miss rates rise. The

compounded uncertainty from cross-node and cross-container execution significantly amplifies the overall squared error.

The MAE curve differs from RMSE. It exhibits a non-monotonic pattern of slight early decline, mid-stage increase, local dip, and final rise. This suggests that the mean absolute error is influenced by both near-term steady states and microbursts. Under moderate quota and moderate preemption, CPU allocation is more stable, and context switching is more controllable. The modeling of local patterns and resource usage by SMB-MFM helps the error converge to a lower level. When entering low-quota and high-preemption scenarios, lightweight interference accumulates into sustained disturbances, causing MAE to rise. However, temporary declines under certain medium-strength settings also show that the model can still capture recoverable periods of stability.

P95 Error shows a distinct sawtooth pattern of spikes, relief, and renewed spikes. This reveals the high sensitivity of tail latency to preemption intensity. High preemption triggers involuntary switches, cache pollution, and kernel contention bursts, which sharply degrade response times in a few time windows. In contrast, moderate preemption or an adequate quota helps suppress tail risks. The cross-scale dependency modeling in TCS-DM can decouple these instantaneous spikes from slow-changing trends. This reduces the drag of spikes on the learning process. However, under extreme preemption, high percentile errors remain, which aligns with the mechanism that tail latency in microservices is often dominated by a few critical path blockages.

The R^2 curve shows a general downward trend with local rebounds. This indicates that when the quota is sufficient or preemption remains moderate, the system dynamics can still be well captured by the model. When the quota becomes too low or preemption too strong, unobserved scheduling noise and workload migration cause distributional drift, reducing model explainability. The combination of multivariate metric representation from SMB-MFM and cross-scale fusion from TCS-DM reveals that when CPU resources and scheduling policies maintain a learnable ratio between slow variables (e.g., trend, seasonality) and fast variables (e.g., preemption, microbursts), R^2 it temporarily improves. Once the system enters a high-interference state, the model requires stronger robust regularization and anomaly awareness to maintain interpretability.

4) Continuous prediction data sensitivity testing under concept drift scenarios (load pattern migration/business peak switching)

This article further presents a continuous prediction data sensitivity test under concept drift scenarios (load pattern migration/business peak switching). The experimental results are shown in Figure 6.

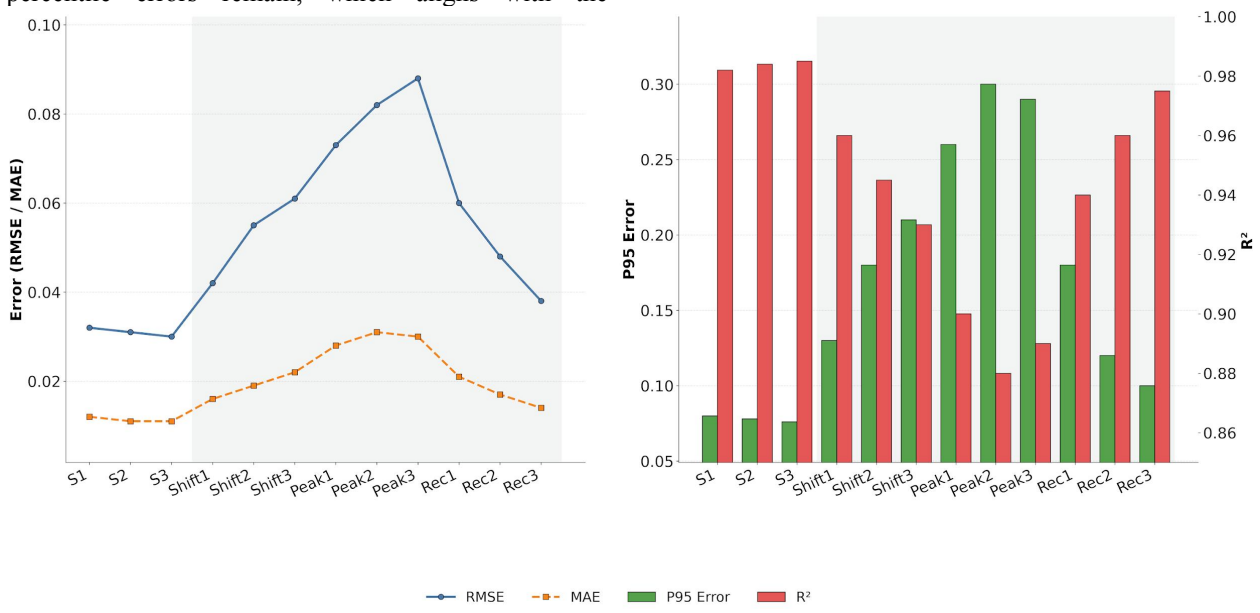

Figure 6. Continuous prediction data sensitivity testing under concept drift scenarios (load pattern migration/business peak switching)

During continuous prediction, as the load pattern shifts from stable phases (S1–S3) to transition phases (Shift1–Shift3), RMSE exhibits a stepwise increase and reaches a peak during high-load switching (Peak1–Peak3), followed by a clear decline in the recovery phase (Rec1–Rec3). This indicates that when the statistical distributions of request arrival rates and resource utilization undergo drift, the prediction error is first amplified in the squared term. Although the cross-scale attention in TCS-DM can smooth short-term uncertainties, it still reveals lag during the early stages of distributional shift. This aligns with the reordering of hot paths in microservice invocation chains under load migration. Reconfigurations of CPU and I/O background usage, queue lengths, and retry strategies change the delay generation mechanism, resulting in temporary mismatches with the parameters learned from previous patterns.

Compared with RMSE, the increase in MAE is more gradual and begins to decrease earlier at the end of the Peak

phase. This shows that the average absolute error adapts more quickly to slow-changing variables and is less sensitive to extreme values. The multivariate modeling of SMB-MFM plays a crucial role in this context. By jointly representing CPU, memory, and network metrics, the model can capture new steady-state trends and recalibrate the baseline more promptly. As a result, MAE declines earlier during the recovery phase. However, in the early stages of transition, MAE still shows a slight increase, suggesting that local patterns-such as burst queueing or cache jitter-can briefly disrupt short-term predictability and must be gradually absorbed through cross-scale fusion.

P95 Error exhibits a clear sawtooth pattern of "spike–relief–spike" during the transition and peak phases. The spike positions align closely with sudden changes in arrival rates, indicating that tail latency is driven primarily by congestion events in a few specific time windows rather than by full-period mean shifts. After separating fast and slow timescales, TCS-DM can decouple spike errors from trend errors. In the later stages of the Peak phase, although mean errors (MAE) start to decline, P95 remains high. This suggests that secondary bottlenecks may persist even after the primary execution paths have adapted. These may include throttling or cold starts in individual service instances, which is consistent with the mechanism that tail latency in microservices is often dominated by low-probability blockages on critical paths.

R^2 Remains high during the stable phases (S1–S3), but drops significantly during the transition and peak phases, and then gradually recovers during the recovery stages. This trend indicates that model interpretability over the data generation process decreases and then restores as concept drift progresses. By combining the insights from previous metrics, it can be observed that when load migration leads to simultaneous changes in metric distributions and temporal dependencies, the cross-scale model requires time to reassign attention weights. During this period, the model's ability to explain overall variance temporarily declines. Once the system reaches a new steady state, the feature pyramid of SMB-MFM and the inter-layer fusion of TCS-DM realign the fast and slow scales, and R^2 rapidly increase again. This temporal sequence-where errors rise first and interpretability follows-illustrates the learnable and stable zones under concept drift in microservice systems. It also provides direct guidance for triggering continuous prediction and online adaptation.

5. Conclusion

This study focuses on response time prediction under a microservice architecture. It proposes a System-Metric-Based Multivariate Feature Modeling (SMB-MFM) mechanism and a Transformer-Based Cross-Scale Dependency Modeling (TCS-DM) mechanism. An end-to-end regression framework is constructed to address the challenges of heterogeneous system metrics, long-path dependencies, and tail latency amplification. Without interfering with business processes, the framework uniformly integrates CPU, memory, I/O, network metrics, and historical response signals. It provides a stable and information-dense foundation for capacity planning, elastic scaling, and service quality assurance.

Existing methods face limitations in aligning and denoising multi-source data, jointly modeling short-term jitters and long-term trends, and maintaining availability under concept drift. These limitations hinder the balance between overall prediction error and tail quantile error. In this work, SMB-MFM explicitly organizes the feature space of multiple metrics, while TCS-DM captures both short- and long-term dependencies through cross-scale attention. Their combination improves the model's capability to handle complex temporal patterns and tail risks.

This framework provides practical value in several application domains. In cloud resource orchestration and auto-scaling, it enables proactive quota adjustments. In AIOps scenarios, it supports dynamic threshold generation, anomaly convergence, and risk assessment. In cost optimization and green computing, it offers measurable signals for joint decision-making across performance, cost, and energy consumption. In edge computing and multi-cloud environments, it improves latency assurance and stability for link-sensitive workloads.

Looking forward, four directions are worth further exploration. First, online learning and drift adaptation can enhance the robustness of continuous prediction. Second, uncertainty estimation and calibration can transform predictive outputs into actionable risk interfaces. Third, structural and causal modeling, incorporating service call graphs and prior knowledge, can improve interpretability and optimization. Fourth, cross-domain transfer and privacy-preserving collaboration can support generalization and compliant model sharing across services. These directions will drive the integration of SMB-MFM and TCS-DM from high-accuracy prediction toward closed-loop decision-making.

However, our evaluation uses a single public dataset and offline training; generalization to unseen service topologies and online drift adaptation remain open. Future work will integrate drift detection and quantile-aware objectives into online training, and couple the predictor with SLO-aware autoscaling or scheduling.

References

[1] Xu H, Liu Y, Xie S, et al. FastPERT: Towards Fast Microservice Application Latency Prediction via Structural Inductive Bias over PERT Networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2025, 39(19): 20787-20795.

[2] Zhang Z, Ramanathan M K, Raj P, et al. {CRISP}: Critical path analysis of {Large-Scale} microservice architectures[C]//2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022: 655-672.

[3] Cai B, Wang B, Yang M, et al. AutoMan: Resource-efficient provisioning with tail latency guarantees for microservices[J]. Future Generation Computer Systems, 2023, 143: 61-75.

[4] Ng N, Souza A, Ali-Eldin A, et al. TailClipper: Reducing Tail Response Time of Distributed Services Through System-Wide Scheduling[C]//Proceedings of the 2024 ACM Symposium on Cloud Computing. 2024: 398-414.

[5] Xin H, Pan R. Self-Attention-Based Modeling of Multi-Source Metrics for Performance Trend Prediction in Cloud Systems[J]. Journal of Computer Technology and Software, 2025, 4(4).

[6] Park J, Choi B, Lee C, et al. Graph neural network-based SLO-aware proactive resource autoscaling framework for microservices[J]. IEEE/ACM Transactions on Networking, 2024, 32(4): 3331-3346.

[7] Bakhtin A, Nyyssölä J, Wang Y, et al. LO2: Microservice API Anomaly Dataset of Logs and Metrics[C]//Proceedings of the 21st International Conference on Predictive Models and Data Analytics in Software Engineering. 2025: 1-10.

[8] Dean, J.; Barroso, L. The Tail at Scale. *CACM*, 2013.

[9] Shabani A, Abdi A, Meng L, et al. Scaleformer: Iterative multi-scale refining transformers for time series forecasting[J]. arXiv preprint arXiv:2206.04038, 2022.

[10] Chen P, Zhang Y, Cheng Y, et al. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting[J]. arXiv preprint arXiv:2402.05956, 2024.

[11] Zhang Y, Yan J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting[C]//The eleventh international conference on learning representations. 2023.

[12] Wu Z, Dong J, Yang H, et al. S2TX: Cross-Attention Multi-Scale State-Space Transformer for Time Series Forecasting[J]. arXiv preprint arXiv:2502.11340, 2025.

[13] Wen Q, Chen W, Sun L, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling[J]. Advances in Neural Information Processing Systems, 2023, 36: 69949-69980.

[14] Du Y, Wang J, Feng W, et al. Adarnn: Adaptive learning and forecasting of time series[C]//Proceedings of the 30th ACM international conference on information & knowledge management. 2021: 402-411.

[15] Zhao L, Shen Y. Proactive model adaptation against concept drift for online time series forecasting[C]//Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1. 2025: 2020-2031.

[16] Bayram F, Ahmed B S, Kassler A. From concept drift to model degradation: An overview on performance-aware drift detectors[J]. Knowledge-Based Systems, 2022, 245: 108632.

[17] Qian W, Zhao H, Chen T, et al. Learning Unified System Representations for Microservice Tail Latency Prediction[J]. arXiv preprint arXiv:2508.01635, 2025.

[18] Xu Y, Ge J, Tang H, et al. System States Forecasting of Microservices with Dynamic Spatio-Temporal Data[J]. arXiv preprint arXiv:2408.07894, 2024.

[19] Ba A, Harsha P, Subramanian C. Leveraging Interpretability in the Transformer to Automate the Proactive Scaling of Cloud Resources[J]. arXiv preprint arXiv:2409.03103, 2024.

[20] Fang Z. A deep learning-based predictive framework for backend latency using AI-augmented structured modeling[J]. Journal of Computer Technology and Software, 2024, 3(7).