Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 4, No. 4, 2025

Self-Attention-Based Modeling of Multi-Source Metrics for Performance Trend Prediction in Cloud Systems

Honghui Xin[1], Ray Pan[2]
[1]Northeastern University, Seattle, USA
[2]Independent Researcher, Seattle, USA
*Corresponding Author: Ray Pan; raypan.research@gmail.com

Abstract: This paper addresses the modeling challenges posed by multi-source heterogeneous metrics in cloud system performance trend prediction and proposes a self-attention-based prediction method. The proposed approach consists of two main components: a Heterogeneous Feature Fusion Module and a Temporal Context-aware Attention Mechanism. The former aligns and weights resource-level, application-level, and environment-level features through unified projection and gating mechanisms. The latter models long-term dependencies among critical time steps in the sequence using a self-attention structure, enabling precise capture of system state evolution trends. The model is trained in an end-to-end manner, demonstrating strong scalability and generalization capability. The experimental evaluation is conducted on the real-world Alibaba Cluster Trace 2018 dataset and includes comparisons with both classical and state-of-the-art methods. The results show that the proposed approach achieves the best performance across all evaluation metrics, including MSE, MAE, RMSE, and R^2. In further sensitivity analysis, the model exhibits robust stability under varying hyperparameter settings, such as input window length, optimizer type, and learning rate. Additionally, the multi-source feature contribution analysis confirms the distinct impacts of different feature categories on prediction performance and highlights the necessity of multi-source information fusion. Overall, the experimental results demonstrate that the proposed method offers significant advantages in terms of accuracy, stability, and practical utility, making it well-suited for performance prediction tasks in complex cloud environments.

Keywords: System performance prediction, self-attention mechanism, multi-source feature fusion, time series modeling.

1. Introduction

With the rapid development of cloud computing technologies, the complexity of modern information systems continues to rise. System scale is expanding, component interactions are becoming more frequent, and performance fluctuations are exhibiting increasingly complex and dynamic characteristics. In the actual operation of cloud platforms, system performance prediction has become a key approach for ensuring efficient resource allocation, improving service stability, and reducing operational costs[1,2]. Especially under the growing influence of features such as multi-tenancy, heterogeneous architecture, and elastic resource scheduling, single-dimensional or static indicators can no longer accurately reflect the system's operating status. Therefore, constructing a performance prediction model capable of integrating multi-source information and understanding contextual dependencies has become both a research hotspot and a technical challenge in the field of cloud computing[3].

However, achieving high-precision performance trend prediction faces numerous challenges. First, various performance metrics in cloud systems exhibit strong nonlinearity and temporal dependencies. Complex interaction dynamics exist among multidimensional indicators such as CPU usage, memory load, I/O rate, and network latency, which are difficult to capture effectively using traditional models. Second, these metrics often originate from different data sources and suffer from issues such as inconsistent time granularity, sampling delays, and distribution heterogeneity.

These problems hinder the ability of prediction models to learn a unified feature representation. In addition, sudden workload changes, scheduling policy adjustments, and external request fluctuations in cloud environments further disrupt the system's operating patterns, imposing stricter requirements on prediction accuracy. Therefore, it is of significant research and practical value to design a prediction mechanism that captures semantic dependencies among multi-source indicators while adapting to dynamic environmental changes[4,5].

In recent years, the self-attention mechanism has demonstrated powerful modeling capabilities in natural language processing and time series analysis, particularly in capturing long-range dependencies and interactions among multidimensional features. Inspired by this, we propose a method for predicting performance trends in cloud systems that integrates self-attention mechanisms with multi-source information fusion[6]. Based on monitoring data from cloud systems, the method introduces a multidimensional feature modeling strategy. It extracts features from different types of indicators through a multi-channel network and employs an improved self-attention structure to dynamically learn correlations and influence weights among the indicators. On this basis, a time-aware contextual modeling module is incorporated to more precisely capture the evolution trends of

system performance. The overall model not only adapts to asymmetric dependencies among indicators but also responds swiftly to sudden fluctuations, thereby significantly enhancing the robustness and accuracy of performance prediction. Compared with existing approaches, the proposed model offers three key advantages[7,8,9]. First, by adopting a strategy that fuses heterogeneous multi-source information, it overcomes the limitation of relying on single indicators in traditional performance prediction and achieves more comprehensive system state awareness. Second, by introducing the self-attention mechanism and dynamic contextual modeling, the model possesses strong semantic representation capabilities and effectively captures temporal dependencies, making it particularly suitable for processing long sequences and high-dimensional inputs. Third, experimental results on real-world cloud platform datasets demonstrate that the proposed method significantly outperforms mainstream baseline models in prediction accuracy across various performance indicators and exhibits superior generalization and real-time adaptability under sudden workload scenarios. Overall, the method provides new insights and practical support for building intelligent and scalable cloud system performance management mechanisms.

In conclusion, this work addresses key challenges in cloud performance trend prediction, including difficulties in multi-source information fusion, insufficient modeling of indicator dependencies, and delayed responses to sudden workloads. We propose a high-performance prediction framework for modeling heterogeneous indicators and complex dependencies. By introducing self-attention mechanisms and dynamic feature modeling techniques, the proposed approach substantially improves the expressiveness and predictive accuracy of system performance modeling. This research not only offers theoretical contributions but also provides technical support for practical applications such as intelligent scheduling, resource optimization, and fault prediction in cloud platforms. In the future, the method can be extended to more complex scenarios such as multi-cloud environments and edge computing, thereby promoting the intelligent development and automated management of cloud infrastructures.

2. Background

2.1 Multi-source features

In cloud system performance prediction tasks, the quality of feature representation directly determines the upper bound of model performance[10,11]. Traditional approaches often rely on single-source data inputs, such as CPU utilization or memory usage, which, although indicative of certain aspects of system operation, are frequently insufficient or incomplete in practical multi-tenant environments. With the advancement of monitoring systems and logging platforms, an increasing number of studies have explored incorporating features from diverse data sources, such as system call sequences, network traffic, disk I/O, task scheduling logs, and even service topology information, in order to construct more comprehensive representations of system states. The integration of multi-source features contributes to improved adaptability of models in complex scenarios and enhances semantic understanding of system behaviors[12].

Existing work primarily adopts feature concatenation or aggregation strategies to integrate multi-source information, typically by mapping various indicators into a unified vector space for combination[13,14]. However, such approaches often overlook the dynamic relationships and temporal dependencies among indicators, making it difficult to capture the latent semantic interactions across data sources. For example, fluctuations in network latency may be caused by increased load on a specific microservice, leading to resource contention, a causal link that cannot be easily inferred from static features alone. Furthermore, different data sources have varying sampling frequencies and temporal resolutions. Naïve concatenation may introduce noise or cause feature redundancy, thereby reducing the generalization capacity of the model. Consequently, incorporating a structured modeling mechanism on top of multi-source features has emerged as a key research challenge[15].

In recent years, some studies have attempted to employ deep learning methods for unified modeling of multi-source features, with structures such as attention mechanisms and graph neural networks demonstrating strong flexibility in handling heterogeneous data. Attention mechanisms can dynamically assign weights to features from different sources, thereby emphasizing the influence of key indicators. Graph neural networks, on the other hand, can be used to model the topological relationships among indicators, enabling cross-dimensional feature interactions. Building on these insights, this study further introduces a self-attention mechanism embedded within a multi-channel feature extraction module. This design enables the model to learn deep semantic associations among multi-source features from a global perspective, significantly enhancing both the expressiveness of feature fusion and the capacity for temporal modeling.

2.2 Self-Attention Mechanism

The self-attention mechanism was originally introduced to address the challenge of modeling long-range dependencies and has demonstrated exceptional feature representation capabilities across various domains, including natural language processing, image recognition, and time series analysis[16,17,18]. Its core principle lies in computing attention weights between arbitrary positions in a sequence to adaptively capture global contextual information, thereby overcoming the gradient vanishing and information forgetting issues commonly encountered in traditional recurrent neural networks when processing long sequences. In system performance prediction tasks, performance metrics exhibit clear long-term dependencies and nonlinear interactions. The introduction of self-attention equips models with enhanced modeling capacity, enabling them to identify the influence of critical moments or key indicators on future trends and thus improving the accuracy and stability of predictions[19].

Existing research has shown that self-attention mechanisms outperform traditional temporal models when handling high-dimensional multivariate time series, particularly in scenarios involving sudden events or pattern shifts in system performance. Compared to feature modeling approaches based on fixed windows or moving averages, self-attention can

dynamically adjust attention distribution to emphasize predictive feature segments. Some studies have further explored multi-head attention structures, which model relationships across different feature subspaces in parallel, thereby enhancing the model's perception of complex system behaviors[20]. However, in practical deployment, self-attention faces challenges such as high computational cost and contextual incompleteness. In particular, when dealing with multi-source heterogeneous inputs, effectively integrating information from multiple attention channels remains a significant challenge.

To address these issues, recent studies have explored combining self-attention with other architectural components, such as integrating it with convolutional networks to capture local patterns, with graph structures to model topological relationships, or introducing sparse attention mechanisms to reduce computational complexity[21]. These explorations have expanded the applicability of self-attention and introduced new approaches to system performance prediction. Building on this foundation, this study proposes an enhanced self-attention module that models semantic-level weights across multi-source input features, ensuring strong representational and generalization capabilities in high-dimensional heterogeneous environments. This design lays a solid foundation for accurate forecasting of system performance trends.

3. method

To address the complex dependencies and dynamic variation patterns among multi-source heterogeneous metrics in cloud systems, this paper proposes a performance trend prediction framework that integrates a self-attention mechanism with multi-channel feature modeling. The goal is to achieve more accurate and robust system performance prediction in high-dimensional, multi-source environments. Based on temporal monitoring data, the method employs an end-to-end deep architecture that combines hierarchical feature extraction, self-attention modeling, and trend decoding modules to capture the nonlinear correlations and evolving patterns among performance indicators. Compared with traditional approaches, the proposed model introduces two key innovations in architectural design. First, a Heterogeneous Feature Fusion Module (HFFM) is proposed to align and semantically integrate system metrics from different sources and dimensions, thereby enhancing the expressive capacity of multi-source information modeling. Second, a Temporal Context-aware Attention Mechanism (TCAM) is designed to dynamically capture critical dependencies within varying temporal windows, thus improving the model's sensitivity and responsiveness to changes in system state trends. The synergy between these two components significantly enhances the adaptability and generalization performance of the prediction model under complex cloud environments. The model architecture is shown in Figure 1.

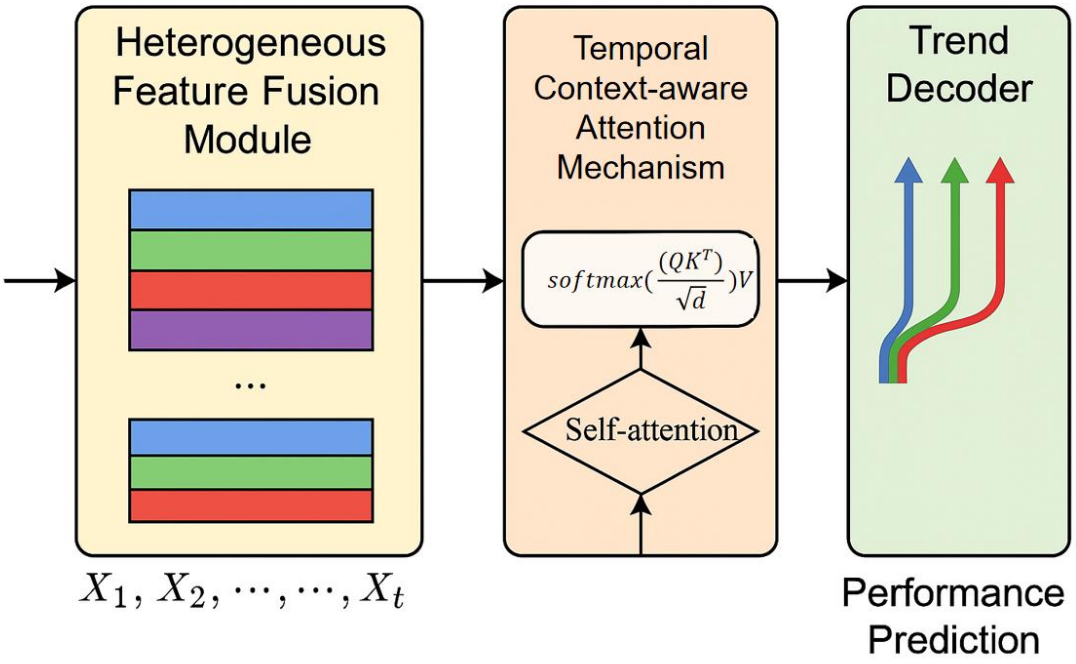

Figure 1. Overall model architecture diagram

3.1 Heterogeneous Feature Fusion Module

To effectively model the complex performance patterns of cloud systems, the proposed framework begins by ingesting a set of heterogeneous time-series features derived from multiple monitoring sources. Its module architecture is shown in Figure 2. These features include resource-level metrics such as CPU and memory usage, application-level indicators like request volume and response time, and environment-level signals such as network latency. By integrating diverse types of temporal data, the framework captures rich contextual information essential for accurate performance trend prediction.

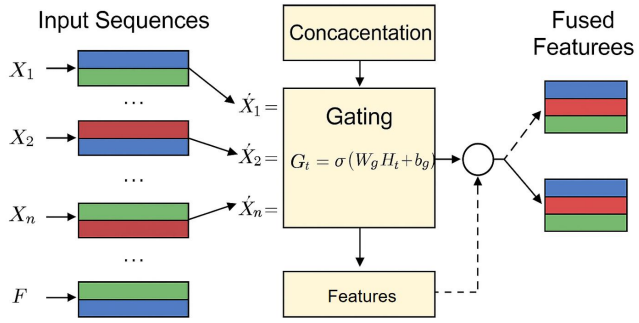

Figure 2. HFFM module architecture

Let the input sequence be $X = \{X_1, X_2, ..., X_t\}$, where each $X_i \in R^{n \times d_i}$ represents a multi-source feature at the i-th time step, n is the number of data sources, and d_i is the feature dimension of the E-th source. These inputs usually include resource layer indicators (such as CPU and memory utilization), application layer signals (such as service latency), and environment layer statistics (such as network I/O). The various features vary significantly in scale, semantics, and dynamics.

In order to solve the problem of feature heterogeneity, this paper linearly maps each feature source and transforms it into a unified latent representation space. Specifically, each type of feature is projected using a trainable linear transformation to map it to a unified dimension:

$$\widetilde{X}_i = \phi_i(X_i) = W_i X_i + b_i$$

Where $W_i \in R^{d' \times d_i}, b_i \in R^{d'}$ is a learnable parameter and d' is a unified potential dimension. This operation ensures that all input features are aligned in the same space, which facilitates subsequent unified modeling.

On this basis, the fusion module concatenates the projection vectors of all sources at each time step and introduces a gating mechanism to dynamically adjust the contribution weights of various features. The concatenated representation is as follows:

$$H_t = [\widetilde{X}_t^{(1)}; \widetilde{X}_t^{(2)}; ...; \widetilde{X}_t^{(n)}]$$

$$G_t = \sigma(W_g H_t + b_g)$$

$$F_t = G_t \otimes H_t$$

$\sigma(\cdot)$ is the Sigmoid activation function, W_g, b_g is a trainable parameter, and $\otimes$ represents element-wise multiplication. This gated fusion mechanism can dynamically adjust the expression strength of features according to the temporal context, thereby enhancing the model's ability to perceive changes in key indicators.

Finally, the fused output of the entire time series is defined as:

$$F = \{F_1, F_2, ..., F_t\}$$

And it is input into the subsequent attention modeling module as a unified high-quality feature representation. This fusion process fully captures the semantic information and intrinsic dependencies of heterogeneous indicators in the system, providing a solid representation foundation for subsequent time series modeling.

3.2 Temporal Context-aware Attention Mechanism

In order to more effectively capture the key dependencies in the evolution of system performance over time, this paper further designs a temporal context-aware attention mechanism (TCAM) based on the fusion features. The goal of this module is to dynamically learn the influence intensity between each time step based on the multi-source features in the historical time series, thereby modeling the potential temporal association. Its module architecture is shown in Figure 3.

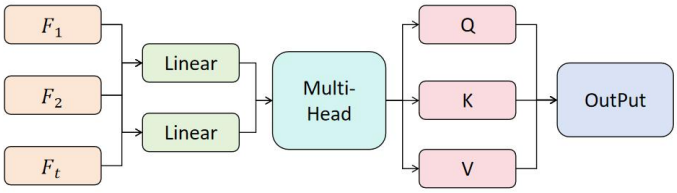

Figure 3. TCAM module architecture

Let the fused sequence be $F = \{F_1, F_2, ..., F_t\}$, where each $F_i \in R^{d'}$ represents the joint feature vector of the i-th time step.

Before performing attention modeling, the feature vector of each time step is first mapped into three groups of representations: query, key, and value to construct the attention weight:

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

Where $W_Q, W_K, W_V \in R^{d' \times d_k}$ is a learnable parameter and d_k is the dimension of attention representation. This step embeds the original time series into the attention calculation space, allowing the model to learn cross-temporal dependencies from it.

The attention weights are calculated using the standard scaled dot product attention mechanism:

$$Attention(Q, K, V1) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The $\text{softmax}(\cdot)$ operation is used to normalize each row to obtain the degree of attention of each time step in the global time series. This operation ensures that the model can capture long-range dependencies and highlight the historical segments that contribute most to future predictions.

To further improve the expressiveness of the model, this paper introduces a multi-head attention mechanism to model the attention representation in different subspaces in parallel:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where h represents the number of attention heads, W_i^Q, W_i^K, W_i^V is the linear transformation matrix corresponding to the i-th head, and W^O is the output mapping matrix. Through parallel modeling of multiple attention subspaces, the model can understand the semantic relationship between temporal contexts from multiple perspectives and improve the stability and robustness of prediction.

4. Experimental Results

4.1 Dataset

This study adopts the Alibaba Cluster Trace 2018 as the system performance dataset for experimentation. Publicly released by Alibaba, this dataset provides comprehensive records of real-world cloud data center operations over an extended period, including resource scheduling, container lifecycles, and machine states. It covers approximately 4,000 physical machines and tens of thousands of containerized tasks, offering high levels of authenticity and representativeness. The data is sampled at a minute-level granularity and includes key performance metrics such as CPU, memory, disk usage, machine status, and resource requests and utilization. This provides a solid foundation for studying resource fluctuations and performance trends in multi-tenant cloud environments.

Unlike synthetic data or static logs, the Alibaba Trace 2018 exhibits distinct characteristics of real-world systems in terms of data density, temporal continuity, and behavioral diversity. The task workloads demonstrate highly dynamic variations, including burstiness, periodicity, and resource contention, which introduce significant modeling challenges for performance prediction. In addition, the dataset contains rich auxiliary information such as task priorities, QoS labels, and user identifiers, offering natural support for multi-source feature modeling and contextual semantic learning. These features greatly enhance the flexibility and generalization capacity of algorithm design.

During the data preprocessing phase, resource usage data is segmented based on container lifecycles, and faulty nodes and outlier samples are removed to ensure stable model training. All features are normalized to the [0, 1] range, and a sliding window approach is applied to construct sequential inputs, allowing the model to learn the trends within continuous performance fluctuations. This dataset serves as a high-quality validation platform for the proposed multi-source feature fusion and temporal attention mechanisms and provides a solid data foundation for future deployment of the algorithm in real-world cloud platforms.

4.2 Experimental setup

To evaluate the effectiveness of the proposed model in system performance prediction tasks, experiments were conducted based on the Alibaba Cluster Trace 2018 dataset within a unified deep learning framework for both training and testing. The original monitoring data was chronologically divided into a training set (70%), a validation set (15%), and a test set (15%) to ensure that the model does not access future information during evaluation. During the modeling process, all input sequences were constructed using a sliding window strategy, with a window size of 60 and a prediction horizon of 10, allowing the model to capture both short-term fluctuations and mid-term trends.

The experiments were conducted on a workstation equipped with a high-performance GPU. The Adam optimizer was used for parameter updates, with mean squared error (MSE) as the loss function, and a learning rate decay strategy was adopted to improve training stability. Table 1 presents the core hyperparameter settings used during model training, including batch size, learning rate, and hidden layer dimensions, to ensure the reproducibility and fairness of the experimental results. The main training configuration is shown in Table 1.

Table 1: Training Configuration

Parameter name	Setting Value
Batch size	128
Initial learning rate	0.001
Optimizer	Adam
Sequence length	60
Prediction horizon	10
Attention heads	4
Hidden dimension	64
Epochs	100

4.3 Experimental Results

1) Comparative experimental results

First, this paper gives the comparative experimental results with other models. The experimental results are shown in Table 2.

Table 2: Comparative experimental results

Method	MSE	MAE	RMSE	R^2
XGBoost[22]	0.0386	0.1473	0.1964	0.831
MLP[23]	0.0321	0.1325	0.1792	0.864
LSTM[24]	0.0268	0.1194	0.1637	0.889
Transformer[25]	0.0243	0.1131	0.1559	0.901
Timemixer[26]	0.0227	0.1088	0.1506	0.912
ITransformer[27]	0.0219	0.1056	0.1480	0.917
Ours	0.0184	0.0972	0.1357	0.931

As shown in the experimental comparison results presented in Table 2, the proposed method achieves the best performance across all evaluation metrics, demonstrating its effectiveness and advantages in system performance prediction tasks. In terms of mean squared error (MSE), our approach achieves a value of 0.0184, significantly outperforming other baseline models. Notably, it shows substantial improvement over

traditional machine learning models such as XGBoost (0.0386) and neural network baselines like MLP (0.0321). This indicates that the proposed model offers superior predictive accuracy, better capturing the complex dynamics of system performance and reducing prediction bias.

Regarding mean absolute error (MAE) and root mean square error (RMSE), which measure prediction stability and deviation magnitude, our method also performs exceptionally well. It achieves an MAE of 0.0972 and an RMSE of 0.1357, which represent reductions of approximately 14.1% and 13.0% compared to the Transformer model, respectively. This suggests that the proposed model not only maintains overall accuracy but also effectively reduces the occurrence of large error events, indicating stronger robustness. In particular, under conditions involving frequent load spikes or resource fluctuations, traditional models often suffer from significant prediction drift. In contrast, our model incorporates multi-source fusion and temporal attention mechanisms to more sensitively capture critical contextual information, resulting in more stable prediction outputs.

From the perspective of the R^2 metric, the proposed method achieves a value of 0.931, which is considerably higher than XGBoost's 0.831 and LSTM's 0.889, indicating a stronger ability to explain performance trends. As a measure of goodness-of-fit, an R^2 value closer to 1 suggests better reconstruction of the true output distribution. The improvement in this metric reflects the model's enhanced capacity for learning temporal feature structures and identifying key dependencies, enabling a more comprehensive mapping between input sequences and prediction targets.

In summary, by comparing against traditional machine learning models (such as XGBoost), classic neural network architectures (MLP and LSTM), and state-of-the-art temporal modeling methods (Transformer, Timemixer, and iTransformer), the proposed approach consistently achieves the best results across all metrics. This demonstrates strong predictive performance and generalization ability. The findings validate the effectiveness of modeling strategies that integrate heterogeneous features and context-aware attention mechanisms, providing solid theoretical and empirical support for future deployment in real-world cloud computing environments.

2) Hyperparameter sensitivity experiment results

Furthermore, the experimental results of hyperparameter sensitivity are given. First, the experimental results of learning rate are given, as shown in Table 3.

Table 3: Hyperparameter sensitivity experiment results (learning rate)

Learning Rate	MSE	MAE	RMSE	R^2
0.004	0.0267	0.1219	0.1634	0.887
0.003	0.0212	0.1067	0.1456	0.919
0.002	0.0195	0.0993	0.1395	0.926
0.001	0.0184	0.0972	0.1357	0.931

Table 3 presents the hyperparameter sensitivity results of the proposed model under different learning rate settings. It can be observed that as the learning rate decreases progressively from 0.004 to 0.001, the model's performance improves consistently across all evaluation metrics. This indicates that the model is sensitive to the learning rate parameter and that an appropriate learning rate selection has a significant impact on optimizing convergence speed and final accuracy. When the learning rate is set to 0.004, the model yields an MSE of 0.0267 and an R^2 of 0.887. Although this suggests a certain level of fitting ability, the prediction error remains relatively high, likely due to the large step size causing the model to skip optimal points or oscillate during training.

As the learning rate is reduced to 0.003 and 0.002, the model's performance improves noticeably, with MSE dropping to 0.0212 and 0.0195, and MAE decreasing to 0.1067 and 0.0993, respectively. This trend demonstrates that moderately lowering the learning rate can guide the model parameters toward the optimum more stably, thereby improving overall prediction accuracy and robustness. Particularly under the 0.002 setting, the model approaches its optimal performance, suggesting that this value serves as a relatively robust candidate that balances training efficiency and accuracy.

At a learning rate of 0.001, the model achieves its best performance, with the MSE reduced to 0.0184 and R^2 improved to 0.931, indicating a more ideal fit. This result suggests that a smaller learning rate facilitates finer gradient updates, allowing the model to capture complex temporal patterns while avoiding overfitting and training instability. It is worth noting that although smaller learning rates may prolong the training process, the improvement in accuracy for this task is substantial enough to justify the additional time cost.

In summary, the learning rate, as a critical optimization hyperparameter, has a direct influence on the performance of the system performance prediction model. The experimental results indicate that gradually decreasing the learning rate from a larger initial value can significantly enhance prediction performance, with the optimal value reached at 0.001. This conclusion not only informs the optimization of the proposed model but also offers a valuable reference for hyperparameter tuning in future related research. Future work may also explore adaptive learning rate strategies to further improve training efficiency and model generalization.

Furthermore, the experimental results of different optimizers are given, as shown in Table 4.

Table 4: Hyperparameter sensitivity experiment results (Optimizer)

Learning Rate	MSE	MAE	RMSE	R^2
AdaGrad	0.0249	0.1123	0.1578	0.896
SGD	0.0317	0.1295	0.1780	0.867
RMSprop	0.0208	0.1024	0.1443	0.922
0.001	0.0184	0.0972	0.1357	0.931

Table 4 presents the impact of different optimizers on model performance under a fixed learning rate setting of 0.001. The results clearly indicate that optimizer selection significantly influences training outcomes in system performance prediction tasks. Among all candidates, the Adam optimizer consistently achieves the best performance across all evaluation metrics, with an MSE of 0.0184, MAE of 0.0972,

RMSE of 0.1357, and R^2 reaching 0.931. This demonstrates Adam's superior overall performance in terms of convergence efficiency, error control, and fitting ability. These findings align with common experience in the deep learning community, where Adam's adaptive learning rate mechanism is known to better accommodate gradient changes and avoid issues such as local minima or vanishing gradients.

In contrast, the SGD optimizer performs the weakest in this task, yielding an MSE of 0.0317 and an R^2 of only 0.867, with significantly higher error compared to other optimizers. This is primarily because SGD relies on a global fixed learning rate and lacks sensitivity to gradient noise and local curvature. As a result, it tends to converge slowly or exhibit unstable fluctuations in complex high-dimensional time series modeling tasks, making it difficult to effectively capture nonlinear dynamic relationships among metrics. Therefore, while SGD may be suitable for certain lightweight tasks, it proves insufficient for multi-source heterogeneous data-driven performance prediction.

The AdaGrad optimizer demonstrates moderate improvement in prediction stability, achieving an MSE of 0.0249 and an R^2 of 0.896, outperforming SGD but still falling short of Adam. While AdaGrad adjusts the learning rate dynamically during training, which allows for faster early-stage convergence, it tends to suffer from overly aggressive learning rate decay in later stages. This behavior limits its ability to further approach the optimal solution during extended training periods, especially when dealing with complex system performance data.

RMSprop, which emphasizes weighted historical gradients, performs better than AdaGrad, reducing the MSE to 0.0208 and increasing the R^2 to 0.922. It approaches Adam in several metrics but still falls slightly behind. This indicates that RMSprop has certain advantages in handling temporally dependent data, though its limited perception of global gradient trends can cause the optimization path to deviate from the global optimum. In summary, these experiments demonstrate the adaptability and stability of the Adam optimizer in system performance prediction tasks, making it the most appropriate choice in the current setting. Future work may explore combined optimizer strategies or meta-learning-based scheduling mechanisms to enhance optimization flexibility across different tasks and training stages.

3) Sensitivity experiment of different input window lengths of the model

Furthermore, this paper also presents a sensitivity experiment on the model with different input window lengths, and the experimental results are shown in Figure 4.

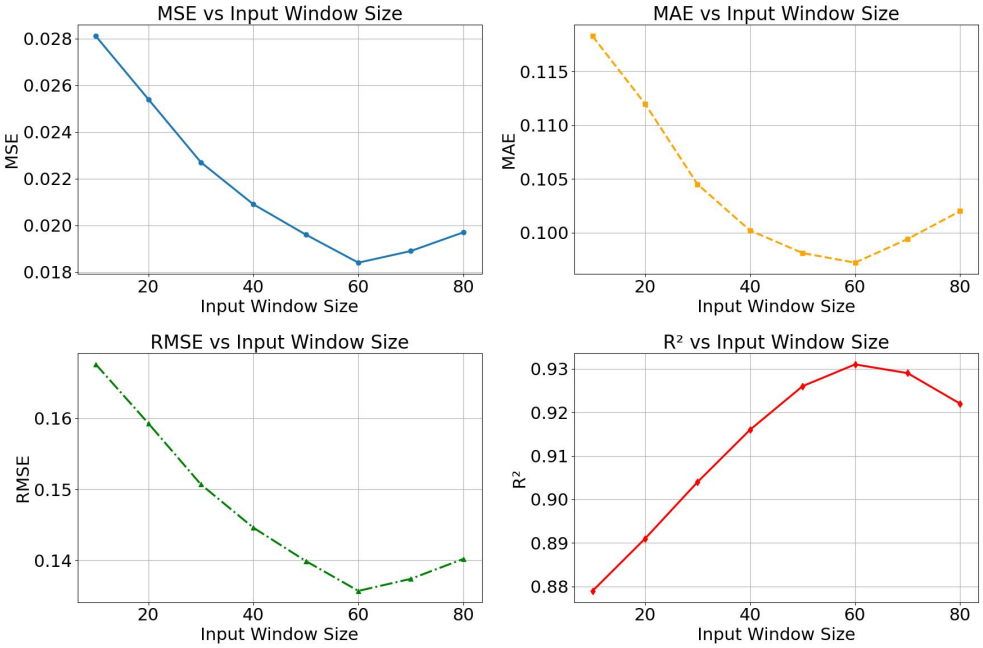

Figure 4. Sensitivity experiment of different input window lengths of the model

Figure 4 illustrates the sensitivity of the model's performance to different input window lengths, with visual analysis conducted using four evaluation metrics: MSE, MAE, RMSE, and R^2. The overall trend shows that as the input window length increases, the model's predictive ability improves within a certain range. However, beyond a specific threshold, performance gains plateau or even slightly decline. This suggests that incorporating additional historical information helps the model capture long-term dependencies in system behavior, but excessively long input windows may introduce redundancy or noise, thereby diverting the model's attention from critical features.

From the MSE and MAE subplots, both metrics reach their optimal values when the input window length is 60, after which a slight rebound is observed. This indicates that a window length of 60 is the optimal setting under the current model

architecture. At window lengths of 10 and 20, the errors are significantly higher, implying that insufficient historical context hampers the model's ability to capture complex performance trends. Between window lengths of 40 and 60, a rapid decline in error is observed, reflecting the model's ability to benefit substantially from additional contextual input in this range.

The trend in RMSE closely aligns with the other two metrics but exhibits greater fluctuation when the window length exceeds 70. This suggests some sensitivity in model stability with respect to certain input lengths. The fluctuations may result from difficulties in temporal alignment or training instability caused by overly long input sequences, highlighting the need to balance representational capacity and stability when designing model input structures. Additionally, the overall RMSE remains below 0.17, further confirming the model's high predictive accuracy.

With regard to the R^2 metric, the model's fitting ability increases progressively with longer input windows, peaking at 0.931 when the window length is 60, followed by a slight decline. This indicates that a moderate increase in historical context enhances the model's capacity to explain performance evolution trends, while excessive expansion may hinder generalization. In summary, an input window length of 60 achieves the best performance across all four metrics, demonstrating an effective trade-off between capturing system dynamics and maintaining model efficiency. This finding provides a strong empirical basis for selecting model parameters in future work.

4) Experimental analysis of the contribution of multi-source features to prediction performance

This paper also presents an experiment to analyze the contribution of multi-source features to prediction performance, and the experimental results are shown in Figure 5.

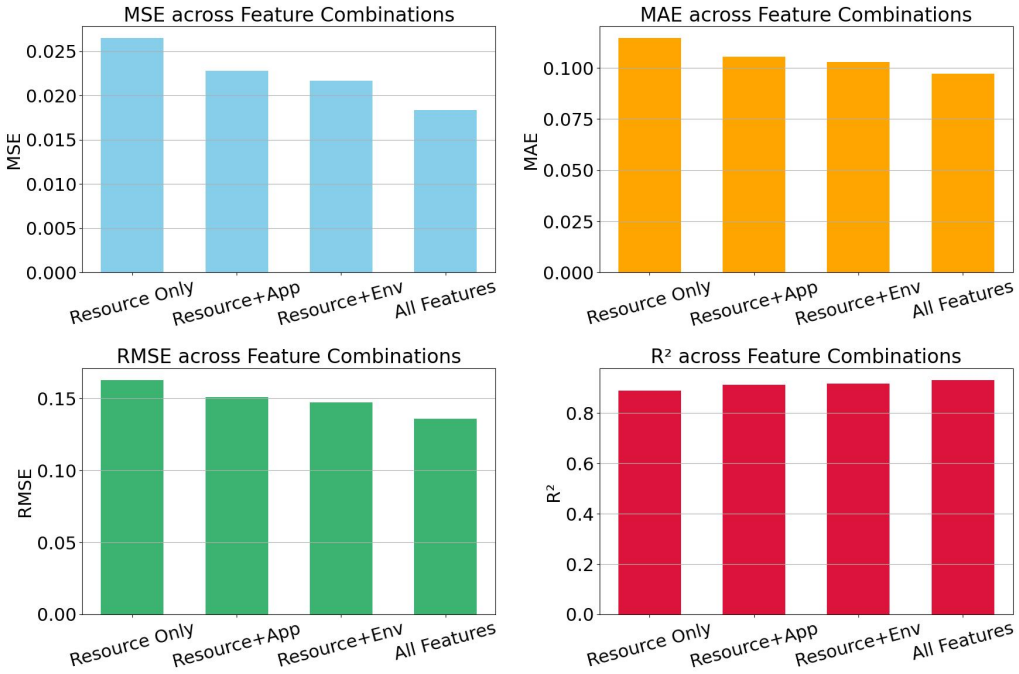

Figure 5. Experimental analysis of the contribution of multi-source features to prediction performance

Figure 5 presents the results of the contribution analysis of multi-source features to model prediction performance, using MSE, MAE, RMSE, and R^2 as evaluation metrics. The horizontal axis represents different feature combination strategies. The experiment progressively introduces features from various sources to evaluate their impact on overall model performance. It can be observed that as feature dimensionality increases, the model's performance improves correspondingly. The most significant enhancement occurs after incorporating environment-level features, confirming the positive contribution of multi-source feature fusion to time series modeling tasks.

In terms of the MSE and RMSE error metrics, using only resource-level features (such as CPU and memory) results in relatively high prediction errors. When application-level features (such as service response time and request volume) are added, error values drop significantly. This indicates that upper-layer business behavior provides valuable representation of underlying system performance. The inclusion of environment-level features (such as network latency and disk I/O) further reduces the errors, suggesting that these macro-level system state indicators provide additional contextual information that helps the model capture more accurate dependency patterns.

The MAE metric shows a similar trend, reflecting enhanced stability as multi-source features are integrated. As shown in the figure, the mean absolute error decreases gradually with the inclusion of more feature types, reaching its lowest point under the “full feature” combination. This indicates that with the support of multi-source information, the model not only

reduces overall error but also responds more consistently to local fluctuations during prediction, demonstrating stronger robustness.

The R^2 metric exhibits a steady upward trend and reaches its highest value of 0.931 when all feature types are combined, indicating the strongest capability in capturing performance trends. This further confirms the complementarity among multi-source features. The joint perception of resource, application, and environment levels significantly enhances the model's explanatory power. In summary, multi-source information fusion plays a critical role in improving the expressiveness and generalization ability of system performance prediction models, providing practical support for building accurate forecasting models in complex system environments.

5. Conclusion

This study addresses the challenge of performance trend prediction in cloud systems and proposes a deep learning framework that integrates self-attention mechanisms with multi-source feature modeling. By introducing a Heterogeneous Feature Fusion Module and a Temporal Context-aware Attention Mechanism, the model effectively extracts key features and constructs global temporal representations in high-dimensional, multi-source, and complex dependency environments. These enhancements significantly improve prediction accuracy and stability. Extensive experimental results demonstrate that the proposed method outperforms mainstream baselines across multiple metrics, validating both the structural design and the superior performance of the model.

Further insights from a series of sensitivity experiments reveal that the model exhibits certain sensitivity to hyperparameters such as input window length, optimizer choice, and learning rate. Appropriate parameter configurations can effectively leverage the strengths of the model architecture. In addition, experiments on multi-source feature fusion highlight the complementary nature of resource, application, and environment-level metrics. Integrating these features not only enhances overall predictive performance but also improves the model's generalization and robustness in complex scenarios. These experimental analyses provide a more comprehensive and nuanced perspective on system modeling, increasing the model's practical utility and engineering value.

Despite the strong results achieved in this work, there remain several directions for further research. For example, the current model focuses primarily on single-task prediction. Future work could explore multi-task learning frameworks to jointly model and predict multiple system performance indicators. Moreover, in multi-tenant heterogeneous environments, incorporating more adaptive mechanisms and cross-domain transfer strategies may further enhance the model's generalization and flexibility in real-world deployment. The interpretability of the model also warrants deeper investigation to improve system engineers' understanding and trust in the prediction results.

In conclusion, this paper presents a forward-looking and practical solution for system performance prediction, with solid theoretical foundations and strong application potential. Future efforts will continue in the directions of high-performance forecasting, interpretable modeling, and cross-scenario generalization to advance the development of intelligent operations and resource scheduling in cloud platforms. We thank all contributors of the experimental platforms and data sources, and we are grateful to the reviewers for their attention and valuable guidance.

References

[1] Yadav M P, Pal N, Yadav D K. Workload prediction over cloud server using time series data[C]//2021 11th international conference on cloud computing, data science & engineering (confluence). IEEE, 2021: 267-272.

[2] Mouine E, Liu Y, Sun J, et al. The analysis of time series forecasting on resource provision of cloud-based game servers[C]//2021 IEEE International Conference on Big Data (Big Data). IEEE, 2021: 2381-2389.

[3] Daraghmeh M, Agarwal A, Manzano R, et al. Time series forecasting using facebook prophet for cloud resource management[C]//2021 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2021: 1-6.

[4] Al-Ghuwairi A R, Sharrab Y, Al-Fraihat D, et al. Intrusion detection in cloud computing based on time series anomalies utilizing machine learning[J]. Journal of Cloud Computing, 2023, 12(1): 127.

[5] Liang Y, Wen H, Nie Y, et al. Foundation models for time series analysis: A tutorial and survey[C]//Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining. 2024: 6555-6565.

[6] Chan K Y, Abu-Salih B, Qaddoura R, et al. Deep neural networks in the cloud: Review, applications, challenges and research directions[J]. Neurocomputing, 2023, 545: 126327.

[7] Dubey A K, Kumar A, García-Díaz V, et al. Study and analysis of SARIMA and LSTM in forecasting time series data[J]. Sustainable Energy Technologies and Assessments, 2021, 47: 101474.

[8] Luo C, Qiao B, Chen X, et al. Intelligent virtual machine provisioning in cloud computing[C]//Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. 2021: 1495-1502.

[9] Yazdanian P, Sharifian S. E2LG: a multiscale ensemble of LSTM/GAN deep learning architecture for multistep-ahead cloud workload prediction[J]. The Journal of Supercomputing, 2021, 77: 11052-11082.

[10] Li X, Li Z, Qiu H, et al. Soil carbon content prediction using multi-source data feature fusion of deep learning based on spectral and hyperspectral images[J]. Chemosphere, 2023, 336: 139161.

[11] Wang S, Li W. GeoAI in terrain analysis: Enabling multi-source deep learning and data fusion for natural feature detection[J]. Computers, Environment and Urban Systems, 2021, 90: 101715.

[12] Wang C, Ma J, Shao J, et al. Non-invasive measurement using deep learning algorithm based on multi-source features fusion to predict PD-L1 expression and survival in NSCLC[J]. Frontiers in immunology, 2022, 13: 828560.

[13] Wang L, Wong L, Li Z, et al. A machine learning framework based on multi-source feature fusion for circRNA-disease association prediction[J]. Briefings in Bioinformatics, 2022, 23(5): bbac388.

[14] Subramanian A S, Weng C, Watanabe S, et al. Deep learning based multi-source localization with source splitting and its effectiveness in multi-talker speech recognition[J]. Computer Speech & Language, 2022, 75: 101360.

[15] Zhang Y, Wang Y. Machine learning applications for multi-source data of edible crops: A review of current trends and future prospects[J]. Food Chemistry: X, 2023, 19: 100860.

[16] Liu, Tianhong, et al. "A hybrid short-term wind power point-interval prediction model based on combination of improved preprocessing methods and entropy weighted GRU quantile regression network." Energy 288 (2024): 129904.

[17] Huang Z, Liang M, Qin J, et al. Understanding self-attention mechanism via dynamical system perspective[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 1412-1422.

[18] Wang Y, Yang G, Li S, et al. Arrhythmia classification algorithm based on multi-head self-attention mechanism[J]. Biomedical Signal Processing and Control, 2023, 79: 104206.

[19] Qian X, Zhang C, Chen L, et al. Deep learning-based identification of maize leaf diseases is improved by an attention mechanism: Self-attention[J]. Frontiers in plant science, 2022, 13: 864486.

[20] Zhu J, Tan Y, Lin R, et al. Efficient self-attention mechanism and structural distilling model for Alzheimer's disease diagnosis[J]. Computers in Biology and Medicine, 2022, 147: 105737.

[21] Zhu W, Wang Z, Wang X, et al. A dual self-attention mechanism for vehicle re-identification[J]. Pattern Recognition, 2023, 137: 109258.

[22] Niazkar M, Menapace A, Brentan B, et al. Applications of XGBoost in water resources engineering: A systematic literature review (Dec 2018–May 2023)[J]. Environmental Modelling & Software, 2024, 174: 105971.

[23] Yu R, Yu W, Wang X. Kan or mlp: A fairer comparison[J]. arXiv preprint arXiv:2407.16674, 2024.

[24] Shiri F M, Perumal T, Mustapha N, et al. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU[J]. arXiv preprint arXiv:2305.17473, 2023.

[25] Tang Y, Wang Y, Guo J, et al. A survey on transformer compression[J]. arXiv preprint arXiv:2402.05964, 2024.

[26] Wang S, Wu H, Shi X, et al. Timemixer: Decomposable multiscale mixing for time series forecasting[J]. arXiv preprint arXiv:2405.14616, 2024.

[27] Liu Y, Hu T, Zhang H, et al. itransformer: Inverted transformers are effective for time series forecasting[J]. arXiv preprint arXiv:2310.06625, 2023.