Artificial Intelligence and Computing Innovations
Intelligence Science Publishing
https://www.ispublishing.com
ISSN: 3070-4944
Vo.5, No.4, 2025

Predictive GPU Memory Modeling in Containerized Cloud Environments Using Self-Supervised Multi-Scale Learning

Xiayuan Liu[1], Kai Gong[2]*
[1]*Northeastern University, Seattle, USA*
[2]*University of Washington, Seattle, USA*
**Corresponding author: kaikaigong221@gmail.com*

Abstract: This paper addresses the complexity and challenges of GPU memory usage prediction in containerized cloud environments and proposes a framework based on self-supervised learning and multi-scale time series modeling. The study first analyzes the dynamic characteristics of GPU memory utilization under multi-tenant and multi-task concurrency and points out that its usage patterns show significant non-stationarity and multi-scale dependencies. To address this, a multi-scale embedding mechanism is designed to unify sequence information at different time granularities, and a self-attention mechanism is applied to capture global dependencies, enhancing the representation of both short-term fluctuations and long-term trends. At the same time, a self-supervised pretraining module is introduced, which extracts latent temporal patterns from large-scale unlabeled data through a masking reconstruction task, reducing the reliance on labeled data and improving model generalization. In the prediction stage, the model uses fused feature representations to perform regression learning of GPU memory usage and adopts a joint optimization objective to improve accuracy while maintaining stability. Comparative and sensitivity experiments are conducted, covering mainstream methods such as LSTM, MLP, 1DCNN, and TimeMixer. The results show that the proposed method outperforms existing models in R^2, MAPE, MAE, and RMSE, and demonstrates robustness across multiple conditions, including window size, missing rate, and workload variations. These findings confirm the practical value and wide applicability of the framework in containerized cloud environments.

Keywords: GPU memory prediction; self-supervised learning; multi-scale modeling; containerized cloud environment

1. Introduction

With the rapid development of containerized cloud environments, GPU resources have become the core driver of high-performance computing and intelligent applications. With the large-scale deployment

of deep learning, scientific computing, and real-time rendering tasks, GPU memory utilization directly determines system throughput and application stability[1]. However, GPU memory usage patterns are dynamically influenced by multiple factors, including task concurrency, model size, batch size, and container orchestration strategies. These influences lead to strong non-stationarity and uncertainty. Without effective prediction of GPU memory usage, resource scheduling may become inefficient, tasks may be interrupted, and overall performance may decline, which limits the potential of applications in cloud environments[2].

In cloud-native architectures, containerization has become the foundation for resource scheduling and isolation. Containers, through lightweight and elastic features, significantly improve resource utilization and cross-platform portability. However, the management and allocation of hardware resources such as GPUs within containers still face challenges. Traditional static configuration or experience-based scheduling methods cannot adapt to the dynamic load changes in multi-tenant environments. This not only results in resource waste but also increases operational risks. A data-driven prediction mechanism can provide forward-looking guidance for GPU memory allocation. This enables fine-grained resource scheduling and intelligent load balancing. Such capabilities are crucial for improving the overall performance of cloud services and enhancing user experience.

In complex containerized environments, GPU memory usage often exhibits cross-task and cross-timescale characteristics. In the short term, memory consumption may fluctuate sharply due to burst requests, batch variations, or data loading delays. In the long term, memory utilization may show periodic patterns or stable trends. These multi-level dynamics make it difficult for simple time series models to capture actual usage behaviors. The key challenge is how to model efficiently while capturing multi-scale dependencies for accurate prediction. A prediction framework that can adapt to complex load scenarios and capture temporal features in a self-adaptive way is essential for improving system resilience and resource utilization[3].

At the methodological level, self-supervised learning provides new opportunities for solving this prediction problem. Unlike traditional methods that rely on large amounts of labeled data, self-supervised learning can automatically extract inherent structures and temporal patterns from massive unlabeled system data. This feature not only reduces the cost of building predictive models but also enhances generalization across different scenarios. In containerized cloud environments, GPU memory usage data is large-scale, diverse, and difficult to label accurately. Self-supervised approaches can uncover hidden patterns within these data and provide a solid foundation for prediction tasks. The integration of self-supervised learning and time series modeling opens a promising research direction for resource management in cloud computing[4].

From an application perspective, GPU memory usage prediction affects not only the efficiency of single tasks but also the service quality and economic benefits of the entire cloud platform. By introducing predictive mechanisms, scheduling systems can allocate memory resources in advance before task execution, avoiding overload and conflicts, while reducing waste caused by idle resources. For cloud service providers, this means improved hardware utilization and reduced operational costs without compromising service stability. For users, it ensures more stable and low-latency computational experiences. This dual value makes GPU memory usage prediction in containerized cloud environments a key element in driving intelligent scheduling, green computing, and sustainable development.

2. Related work

Existing studies on GPU resource prediction can be divided into two categories. The first relies on traditional statistical and time series methods. The second is based on deep learning. The former mainly employs autoregressive models, moving average models, or state space methods to model the historical series of GPU memory and utilization. These methods provide interpretability and ease of implementation. However, they show limited accuracy under highly non-stationary workloads and

complex cloud environments. They also struggle with multi-scale features and nonlinear dependencies, making it difficult to meet the real-time and robustness requirements of containerized systems[5].

With the integration of cloud computing and artificial intelligence, deep learning methods have become mainstream for resource prediction. Convolutional neural networks, recurrent neural networks, and attention-based models are widely applied to GPU utilization and memory consumption modeling. These approaches can automatically learn complex nonlinear relationships and handle large-scale data efficiently, leading to significant improvements in prediction accuracy. However, most studies rely heavily on labeled data. In real containerized environments, GPU usage data is large in scale but often lacks explicit labels. This limits the practical deployment of deep learning methods. In addition, many models have complex structures, which demand high training resources and increase inference costs, creating challenges for real-world applications[6].

In recent years, resource prediction in cloud environments has paid increasing attention to multi-dimensional feature modeling. Some studies attempt to combine CPU, memory, and network bandwidth into multimodal or multitask prediction frameworks. These methods provide a more comprehensive view of system operation and support collaborative scheduling across resources. However, GPU memory has unique characteristics as a critical hardware resource. Its prediction requires attention to specific load patterns and allocation rules. General multimodal approaches may fail to capture these unique properties. Designing frameworks that focus specifically on GPU prediction, therefore, remains a pressing challenge[7].

In methodological evolution, self-supervised learning has emerged as a promising direction for resource prediction. By designing pretraining tasks, self-supervised learning can capture hidden temporal patterns and semantic information without additional labels. This is particularly important in containerized cloud environments where data is massive, features are complex, and patterns change frequently. Combining self-supervised learning with GPU memory prediction reduces reliance on manual labels. It also enhances generalization and adaptability across different tasks. This provides an important foundation for building intelligent scheduling and resource management systems.

Recent studies on backend observability, anomaly reasoning, and scheduling-oriented prediction further broaden the methodological foundation for GPU memory usage prediction in containerized clouds. Structure-aware root cause localization for microservice backends shows how joint graph representation learning and causal inference can model dependency structures among services and resources, which supports topology-sensitive analysis of container resource behavior[8]. Behavior representation learning for reliable monitoring of AI inference backends emphasizes the value of learning compact operational states from runtime signals for stable performance supervision[9]. Consistency-aware unified modeling for multi-source system anomaly detection indicates that heterogeneous metrics, logs, and system events can be fused under a unified representation to improve robustness in dynamic environments[10]. Multi-scale convolution and adaptive attention mechanisms used for backend latency forecasting provide a temporal design reference for capturing bursty and long-range workload changes[11]. Secure parameter aggregation and distributed anomaly detection in cross-cloud data center networks extend the discussion to privacy-preserving and distributed resource intelligence[12]. Reinforcement learning and meta-learning based backend service scheduling connects predictive modeling with adaptive scheduling decisions under changing workloads[13], while fine-grained anomaly classification through structural dependency learning and version-aware feature alignment strengthens the interpretation of abnormal resource states during system evolution[14]. Together, these studies support a prediction framework that combines multi-source monitoring, temporal representation learning, structural dependency awareness, and scheduling-oriented deployment.

3. Proposed Approach

In the proposed framework, GPU memory usage prediction is modeled as a time series regression problem. Let $X = \{x_1, x_2, ..., x_T\}$ be the input time series, where x_T represents the multidimensional system features collected at time t, including GPU utilization, batch size, and number of containers. The goal is to predict GPU memory usage $\hat{y}_{t+\tau}$ at a future time, where τ is the prediction step size. The overall prediction process can be formalized as a function mapping:

$$\hat{y}_{t+\tau} = f_\theta(x_1, x_2, \ldots, x_t) \tag{1}$$

Where f_θ represents a prediction model with learnable parameters θ. Through this modeling approach, the framework can capture both short-term fluctuations and long-term trends, thereby achieving multi-scale prediction capabilities. The overall model architecture is shown in Figure 1.

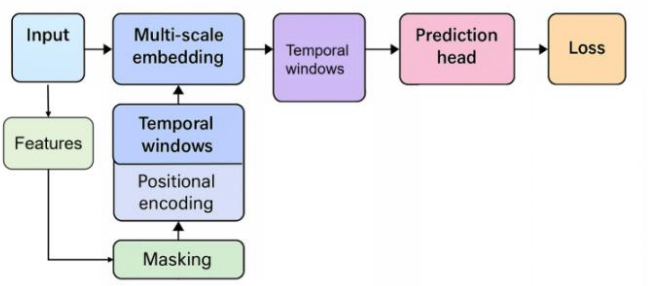


Figure 1. Overall model architecture

In the sequence representation learning stage, a multi-scale embedding mechanism is introduced to map the input sequence into representation spaces of different time granularities. For any time scale s, its embedding representation can be defined as:

$$h_t^{(s)} = Embed_s(x_{t-s+1}, \ldots, x_t) \tag{2}$$

Where $Embed_s(\cdot)$ represents the embedding function at scale s, which is used to capture the local pattern within the time window. Subsequently, the embeddings at all scales are concatenated to obtain a comprehensive representation:

$$H_t = \bigoplus_{s \in S} h_t^{(s)} \tag{3}$$

Where $\oplus$ represents the vector concatenation operation and S represents the multi-scale set. This mechanism ensures that different time-dependent features can be uniformly represented and modeled.

In the process of representation modeling, a dependency capture method based on the self-attention mechanism is introduced. For the comprehensive representation H_t at time t, its attention weight is defined as:

$$\alpha_{t,j} = \frac{\exp((H_t W_Q)(H_j W_K)^T / \sqrt{d_k})}{\sum_{k=1}^{t} \exp((H_t W_Q)(H_k W_K)^T / \sqrt{d_k})} \tag{4}$$

Where $W_Q, W_K \in R^{d \times d_k}$ is the projection matrix of the query and the key, respectively. The representation of global dependency enhancement is obtained by weighted summation:

$$z_t = \sum_{j=1}^{t} \alpha_{t,j}(H_j W_V) \tag{5}$$

Where W_V is the projection matrix of the values. This mechanism enables the model to flexibly capture long-range dependencies in the sequence.

In terms of training objectives, a combination of self-supervised pre-training and regression prediction is adopted. In the pre-training stage, pseudo labels are constructed through a mask mechanism, and the reconstruction loss is defined as:

$$L_{pre} = \frac{1}{|M|} \sum_{t \in M} \|x_t - \hat{x}_t\|_2^2 \tag{6}$$

Where M is the set of masked locations, and $\hat{x}_t$ is the reconstructed output of the model. In the prediction phase, regression learning of future GPU memory is achieved by minimizing the mean square error:

$$L_{pred} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{7}$$

The final optimization objective combines the two losses and is defined as:

$$L = \lambda L_{pre} + (1 - \lambda) L_{pred} \tag{8}$$

Where $\lambda \in [0,1]$ is the balance coefficient. This design ensures that the model can effectively complete the prediction task while making full use of unlabeled data.

4. Performance Evaluation

4.1 Dataset

In this study, the Google Cluster GPU Trace Dataset is selected as the primary data source. This dataset is composed of GPU usage logs collected from real cloud environments. It spans a long period of operation and includes rich features such as task submission, scheduling, GPU memory utilization, task duration, and multi-tenant concurrency information. Such a dataset provides a comprehensive representation of GPU resource dynamics in containerized scenarios, ensuring that prediction research carries both practical significance and engineering value.

The dataset is characterized by high-frequency sampling and large-scale samples. It records tens of thousands of task instances and resource usage trajectories in cloud data centers. Compared with small-scale experimental data, it better reflects complex patterns such as resource competition, burst fluctuations, and periodic regularities under multi-tenant conditions. These properties naturally provide multi-scale temporal features. They enable models to perform joint modeling across short-term and long-term dependencies, which improves the adaptability of the prediction framework.

In addition, the openness and authority of the Google Cluster GPU Trace Dataset provide reproducibility and comparability for research. Since it has been widely used in the field of resource scheduling and prediction, adopting this dataset not only ensures alignment with realistic scenarios but also enables fair comparison with existing methods. This highlights the innovative value and application potential of the proposed framework for GPU memory usage prediction in containerized cloud environments.

4.2 Experimental Results

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

Table 1. Comparative experimental results

Method	R^2	MAPE	MAE	RMSE
LSTM[15]	0.842	0.0937	7.86	10.21
MLP[16]	0.857	0.0892	7.43	9.87
1DCNN[17]	0.869	0.0821	6.95	9.34
TimeMixer[18]	0.884	0.0764	6.41	8.77
Ours	0.913	0.0692	5.86	8.12

From the comparison results in Table 1, it can be observed that traditional sequence modeling methods have certain limitations in predicting GPU memory usage in containerized cloud environments. Recurrent networks such as LSTM can capture temporal dependencies to some extent. However, because they cannot model both short-term and long-term features simultaneously, their R^2 value reaches only 0.842, and error indicators remain high. This shows that in highly dynamic and non-stationary resource usage scenarios, single sequence modeling is not sufficient to achieve high prediction accuracy.

In contrast, MLP and 1DCNN show advantages in capturing nonlinear relationships and local patterns. They outperform LSTM in MAPE, MAE, and RMSE, indicating that exploiting local features can reduce prediction bias. However, these methods cannot still model global dependencies and cross-scale features. As a result, the improvement in prediction accuracy is limited, and they cannot fully adapt to the complexity of multi-task concurrency and load fluctuations in containerized environments.

The performance of TimeMixer further demonstrates the value of multi-scale dependency modeling. Its R^2 increases to 0.884, while error metrics decrease significantly. This indicates that introducing the time-mixing mechanism builds stronger connections between short-term and long-term features. In this way, the model provides a more comprehensive description of the dynamic characteristics of GPU memory usage and improves adaptability in complex environments.

The proposed model achieves the best results across all metrics. The R^2 reaches 0.913, MAPE decreases to 0.0692, and MAE and RMSE are reduced to 5.86 and 8.12, respectively. This shows that the framework has stronger capabilities in capturing non-stationary, multi-scale, and nonlinear dependencies. It can effectively enhance prediction accuracy and stability. For containerized cloud environments, this means more reasonable GPU memory allocation and scheduling, leading to improved overall resource utilization and service quality.

This paper also presents an experiment on the sensitivity of multi-scale window length to R^2, and the experimental results are shown in Figure 2.

From the figure, it can be seen that the change in multi-scale window length has a significant impact on the R^2 metric. As the window length increases gradually from 2 to 14, the coefficient of determination for GPU memory usage prediction shows a steady upward trend. This indicates that longer time windows provide the model with richer contextual information, which helps capture long-term dependency structures in resource utilization and improves overall predictive fitting ability.

With shorter window lengths, the model can only use limited historical information for prediction and therefore relies more on local pattern learning. In this case, the model often fails to effectively capture the periodic and trending characteristics of GPU memory usage in containerized cloud environments, which leads to biased results. As the window length increases, the model gradually acquires the ability to model across time scales, enabling it to perform more stably under complex load fluctuations.

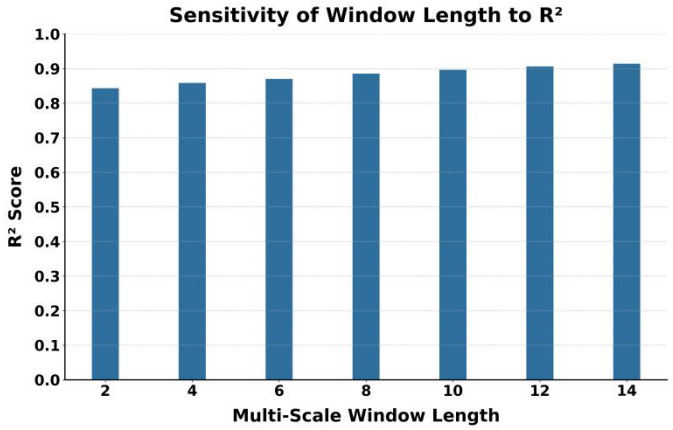


Figure 2. Experiments on the sensitivity of multi-scale window length to R^2

It is worth noting that as the window length continues to increase, the improvement in R^2 becomes less pronounced. This shows that although larger windows enhance the ability of the model to capture long-term dependencies, the marginal contribution of additional context information to prediction performance becomes limited once a certain scale is reached. This finding suggests that in practical deployment, it is necessary to balance prediction accuracy with computational cost to choose an appropriate window length.

Overall, these results confirm the necessity of multi-scale mechanisms in GPU memory usage prediction. By setting the window length appropriately, the model can effectively integrate short-term fluctuations with long-term trends. This improves the application value of prediction in containerized cloud environments. It not only enhances the reliability of prediction but also provides more accurate evidence for subsequent resource scheduling and load balancing.

This paper further presents a sensitivity experiment on the missing data rate to MAPE, and the experimental results are shown in Figure 3.

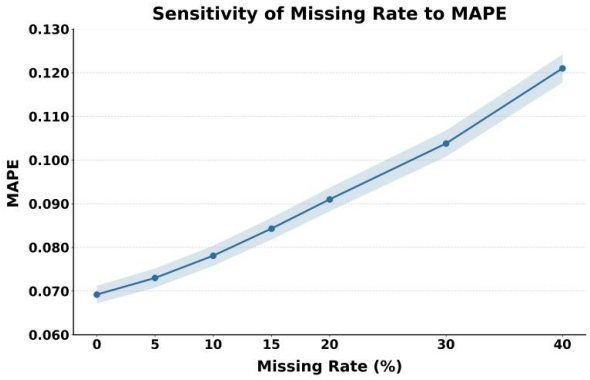


Figure 3. Sensitivity experiment of missing data rate to MAPE

From Figure 3, it can be observed that as the data missing rate increases, the MAPE metric shows a steady upward trend. This indicates that missing information has a direct negative impact on model accuracy in GPU memory usage prediction. When the missing rate is low, the model can still maintain reasonable predictive ability. However, as the missing proportion grows, prediction errors are significantly amplified, showing that the model has a strong reliance on complete data.

At lower missing rates, the increase in MAPE is relatively moderate. This suggests that the model can still use the available data features to preserve a certain level of robustness. In this stage, the model benefits from temporal modeling and self-supervised representation learning, which help compensate for

the lack of information caused by local gaps. As the missing rate continues to rise, especially beyond 20 percent, the MAPE curve increases more sharply. This reflects that once missing information accumulates to a certain degree, it produces a systematic impact on prediction results.

This trend highlights the importance of data integrity in containerized cloud environments. GPU memory usage sequences contain complex dynamic features. An increase in missing values disrupts the continuity and completeness of temporal patterns, making it difficult for the model to capture multi-scale dependencies. As a result, when data quality declines, the predictive advantage of the model is weakened, leading directly to lower accuracy and stability. For resource scheduling and load balancing, this represents a critical risk factor.

Overall, the experimental results show that the missing rate affects not only point-level prediction accuracy but also the reliability of the entire predictive pattern. In practical applications, it is necessary to design more robust predictive models. At the same time, effective mechanisms for handling missing values should be integrated into data collection and storage processes. This can reduce the negative impact of missing information on prediction performance. These findings provide important references for future research on data recovery methods and robust modeling strategies.

5. Conclusion

This paper focuses on the problem of GPU memory usage prediction in containerized cloud environments and proposes a framework based on self-supervised learning and multi-scale temporal modeling. The study analyzes the dynamic characteristics and uncertainty of resource utilization in cloud environments. To address the non-stationarity and multi-scale dependencies of GPU memory usage patterns, a method is developed to capture both short-term and long-term features effectively. With this method, the model shows strong ability in sequence representation, global dependency modeling, and feature fusion. It provides both theoretical and technical support for resource prediction in complex environments. The results demonstrate that the proposed framework improves prediction accuracy and stability, and also verify the potential of self-supervised mechanisms in modeling large-scale system operation data.

From an application perspective, this study offers a practical prediction-driven solution for intelligent scheduling in cloud environments. In real containerized deployments, GPU resources play a central role, and prediction accuracy directly affects task execution efficiency and overall service quality. The proposed method can anticipate GPU memory consumption in advance, providing forward-looking evidence for resource allocation and task scheduling. This helps avoid operational risks caused by overload or shortage. Such capability not only increases system throughput and stability but also reduces resource waste, aligning with the current goals of green computing and efficient cloud computing.

Furthermore, this research has significant implications for related application domains. In GPU-intensive scenarios such as artificial intelligence training, scientific computing, and high-performance rendering, resource demands are often highly complex. Traditional static configuration cannot meet the dynamic requirements of these tasks. The proposed framework enables fine-grained prediction in multi-tenant and multi-task environments and provides useful references for resource scheduling strategies in different applications. It improves task execution efficiency, enhances overall resource utilization, and strengthens service capability at the platform level. It also offers valuable insights for large-scale GPU cluster deployment.

Finally, this study lays the foundation for future exploration. On the one hand, the prediction model provides a scalable framework for more complex research on joint scheduling across multiple resources, including CPU, memory, and network. On the other hand, with the increasing adoption of large-scale data centers, reducing training costs and inference latency while maintaining prediction accuracy will become a critical direction. By integrating data recovery mechanisms, robust modeling methods, and distributed training strategies, the proposed approach has the potential to demonstrate value in larger and more

complex scenarios. This will drive cloud resource management toward higher levels of intelligence and efficiency.

References

[1] Cui G, Hu T, Zhang W, et al. MMTransformer: a multivariate time-series resource forecasting model for multi-component applications[J]. Scientific Reports, 2025, 15(1): 29740.

[2] Tripathi S, Shibu A, Misra R, et al. Optimized resource management in containerized clouds via Hierarchical Autoregressive Network based workload prediction[J]. Applied Soft Computing, 2025: 113781.

[3] Xu M, Song C, Wu H, et al. esDNN: deep neural network based multivariate workload prediction in cloud computing environments[J]. ACM Transactions on Internet Technology (TOIT), 2022, 22(3): 1-24.

[4] Xu F, Xu J, Chen J, et al. igniter: Interference-aware gpu resource provisioning for predictable dnn inference in the cloud[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 34(3): 812-827.

[5] Saxena D, Singh A K. A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center[J]. Neurocomputing, 2021, 426: 248-264.

[6] Guruge P B, Priyadarshana Y. Time series forecasting-based kubernetes autoscaling using facebook prophet and long short-term memory[J]. Frontiers in Computer Science, 2025, 7: 1509165.

[7] Turin G, Borgarelli A, Donetti S, et al. Predicting resource consumption of kubernetes container systems using resource models[J]. Journal of Systems and Software, 2023, 203: 111750.

[8] Wang L, Chang C Y. Structure-Aware Root Cause Localization for Microservice Backends via Joint Graph Representation Learning and Causal Inference[J]. 2024, 3(1).

[9] Ding T, Yuan Y. Behavior Representation Learning for Reliable Monitoring of AI Inference Backends[J]. 2024, 4(1).

[10] Yang Y, Nagel M, Li A. Consistency-Aware Unified Modeling for Multi-Source System Anomaly Detection[J]. 2025, 4(4).

[11] Pi S. Intelligent Backend Latency Forecasting with Multi-Scale Convolution and Adaptive Attention Mechanisms[J]. 2024, 3(5).

[12] Zhang S. Secure Parameter Aggregation and Distributed Anomaly Detection in Cross-Cloud Data Center Networks[J]. 2024, 3(4).

[13] Xue F, Zhu G, Taha J. Intelligent Backend Service Scheduling through the Integration of Reinforcement Learning and Meta-Learning[J]. 2025, 4(6).

[14] Liu W. Fine-Grained Backend Anomaly Classification through Structural Dependency Learning and Version-Aware Feature Alignment[J]. 2024, 4(4).

[15] Saikhu A. CPU Usage Forecasting for Load Balancing in Kubernetes Using LSTM: A Synthetic Traffic Simulation Approach[C]//2025 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT). IEEE, 2025: 277-282.

[16] Gördén A. Predicting resource usage on a kubernetes platform using machine learning methods[J]. 2023.

[17] Leka H L, Fengli Z, Kenea A T, et al. Workload prediction of virtual machines using integrated deep learning approaches over cloud data centers[M]//Human-Centric Smart Computing: Proceedings of ICHCSC 2022. Singapore: Springer Nature Singapore, 2022: 55-65.

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