AERO: Adaptive Edge-Cloud Orchestration with a Sub-1K-Parameter Forecasting Model

Berend J.D. Gort, Godfrey M. Kibalya, and Angelos Antonopoulos

Abstract

Effective resource management in edge-cloud networks is crucial for meeting Quality of Service (QoS) requirements while minimizing operational costs. However, dynamic and fluctuating workloads pose significant challenges for accurate workload prediction and efficient resource allocation, particularly in resource-constrained edge environments. In this paper, we introduce **AERO** (Adaptive Edge-cloud Resource Orchestration), a novel lightweight forecasting model designed to address these challenges. AERO features an adaptive period detection mechanism that dynamically identifies dominant periodicities in multivariate workload data, allowing it to adjust to varying patterns and abrupt changes. With fewer than 1,000 parameters, AERO is highly suitable for deployment on edge devices with limited computational capacity. We formalize our approach through a comprehensive system model and extend an existing simulation framework with predictor modules to evaluate AERO’s performance in realistic cloud-edge environments. Our extensive evaluations on real-world cloud workload datasets demonstrate that AERO achieves comparable prediction accuracy to complex state-of-the-art models with millions of parameters, while significantly reducing model size and computational overhead. Simulations show that AERO improves orchestration performance, achieving up to 13% reduction in energy consumption and 67% reduction in response times compared to reactive approaches. Our live deployment experiments further validate these findings, demonstrating that AERO consistently delivers superior performance. These results highlight AERO as an effective solution for improving resource management and reducing operational costs in dynamic cloud-edge environments.

# INTRODUCTION

computing and cloud-native architectures are reshaping the computational landscape by bringing computational resources closer to end-users, enabling low-latency services  (Shang et al. 2023). This shift enhances application performance but introduces challenges in optimizing resource utilization due to highly dynamic and fluctuating workloads in edge-cloud environments   (Collet et al. 2023; Lu et al. 2017). Efficient resource management is essential to meet Quality of Service (QoS) requirements while minimizing operational costs  (Lu et al. 2017).

Accurate workload prediction at the container level has become crucial for proactive orchestration and resource management in edge-cloud systems  (Guo et al. 2019). The widespread adoption of microservice architectures has introduced new challenges, as applications are decomposed into numerous containers with intricate dependencies and resource consumption patterns  (Saxena et al. 2023). This architectural shift demands more granular predictions for effective container-specific elasticity operations. However, traditional prediction models struggle to capture these complex dynamics, as containerized microservices exhibit unpredictable demand fluctuations and frequent migrations across heterogeneous edge-cloud environments  (Ouyang et al. 2023; Benidis et al. 2022).

The state-of-the-art in cloud workload prediction can be categorized into five main approaches  (Kumar and Singh 2018; Tang 2019). Evolutionary Learning methods integrate neural networks with evolutionary optimization techniques  (Kumar, Singh, and Buyya 2021). Deep Learning approaches form the predominant category, particularly focusing on temporal dependencies in cloud workloads  (Q. Zhang et al. 2018; W. Zhang et al. 2017), though they face challenges with long-range predictions and rapid workload changes  (Hochreiter 1998). Hybrid Learning combines multiple algorithms to overcome individual limitations  (Kardani-Moghaddam, Buyya, and Ramamohanarao 2020; Karim et al. 2021), while Ensemble Learning leverages multiple predictive models for improved accuracy  (N. Singh and Rao 2014; Iqbal et al. 2019). Finally, Quantum Learning has emerged as a promising direction for handling the complexity of modern cloud environments  (A. K. Singh et al. 2021). However, most of these approaches rely on classical architectures like Recurrent Neural Networks (RNNs), while the machine learning literature has been rapidly evolving with newer architectures.

Recent machine learning advancements have introduced architectures like Graph Neural Networks (GNNs)  (Yi et al. 2024), temporal Convolutional Neural Networks (CNNs)  (donghao and xue 2024), and Transformer-based models  (Gort et al. 2025). These models excel in capturing complex temporal patterns but are often computationally intensive due to their large number of parameters, making them unsuitable for resource-constrained edge devices  (Meuser et al. 2024). For example, models like WGAN  (Arbat et al. 2022) and Pathformer  (P. Chen et al. 2024) contain millions of parameters, requiring substantial computational resources and energy consumption, which is impractical for many edge environments  (Kong et al. 2022).

To mitigate these limitations, lightweight Multi-Layer Perceptron (MLP) architectures   (Ekambaram et al. 2023) have been proposed. One of these models, SparseTSF   (Lin et al. 2024) is based on sparse multilayer perceptrons (MLPs) and is designed for fast inference with minimal resource consumption, making it suitable for edge deployment. It achieves performance comparable to larger models while significantly reducing model size. However, SparseTSF assumes a fixed periodicity in workload data, limiting its adaptability in environments where workload patterns exhibit dynamic periodicities or abrupt changes.

Models assuming fixed periodicities may fail to capture sudden changes or multiple overlapping patterns   (Lin et al. 2024), leading to decreased prediction accuracy and suboptimal resource allocation. There is a critical need for lightweight models that can adaptively adjust to changing workload data to improve prediction accuracy and resource efficiency  (Fan et al. 2019).

In this paper, we introduce **AERO** (**A**daptive **E**dge-cloud **R**esource **O**rchestration), a novel forecasting model designed to address the challenges of dynamic workload prediction in resource-constrained environments. AERO is a lightweight model with fewer than 1,000 parameters, making it suitable for deployment on edge devices with limited computational capacity. It incorporates an adaptive period detection mechanism that dynamically identifies dominant periods in workload data, allowing it to adjust to varying periodic patterns. This adaptability enhances prediction accuracy and resource allocation efficiency compared to models assuming fixed periodicities like SparseTSF.

To validate the effectiveness of AERO, we leveraged the **COSCO** (**C**oupled **S**imulation and **C**ontainer **O**rchestration) framework  (Tuli et al. 2021), which provides a realistic emulation environment for service orchestration. COSCO accurately replicates real-world container behavior, including migration patterns, resource allocation, and system dynamics, enabling thorough evaluation of workload predictors without physical infrastructure costs. The framework’s creators have extensively validated it against real fog computing environments, demonstrating high correlation between simulated and actual performance metrics  (Tuli et al. 2021). As a novel contribution, we extend COSCO with a prediction interface.

Our key contributions are:

* We propose AERO, a lightweight adaptive model for edge-cloud workload prediction that effectively handles workloads with multiple or changing periodicities in multivariate data.
* We demonstrate that AERO achieves comparable or better accuracy than state-of-the-art models while significantly reducing model size and computational overhead, suiting resource-constrained environments.
* We extend the COSCO simulation framework with a prediction interface.
* Through evaluations on real-world datasets and simulations, we demonstrate AERO’s ability to improve resource allocation and reduce operational costs in edge-cloud environments.

The remainder of this paper is organized as follows: We first present the notation used throughout the paper in Table . Section  reviews related work in time series forecasting. Section  formulates the workload prediction problem. Section  details AERO’s architecture and implementation. Section  evaluates AERO against existing models. Section  presents simulation results, and Section  concludes with future directions.

Table 1: Notation Summary

| **Symbol** | **Desc.** | **Symbol** | **Desc.** |
| --- | --- | --- | --- |
|  | Nodes |  | Latency |
|  | Node |  | Norm. time |
|  | Containers |  | Objective |
|  | Container |  | weight |
|  | Jobs in |  | Time step |
|  | Job ID |  | Done jobs |
|  | Arrival time |  | End time |
|  | Resources |  | State |
|  | Power |  | Kernel size |
|  | Max power |  | Period |
|  | Energy |  | Prediction |
|  | Loss |  | Parameters |
|  | Workload |  | Features |
|  | Time steps |  | Predictor |
|  | Context |  | Horizon |

# RELATED WORK

Traditional cloud workload prediction approaches can be categorized into five main methodologies: Evolutionary Learning, Deep Learning, Hybrid Learning, Ensemble Learning, and Quantum Learning  (Kumar and Singh 2018; Tang 2019). However, these traditional methods have not kept pace with the rapid advancements in general-purpose time series forecasting architectures based on modern machine learning techniques.

Recent machine learning breakthroughs have introduced sophisticated model architectures that surpass these traditional methods in capturing complex temporal patterns  (Mendis, Wickramasinghe, and Marasinghe 2024). These state-of-the-art architectures can be broadly categorized into several key areas.

RNNs  (Lai et al. 2018; Jhin, Kim, and Park 2024; Y. Li, Xu, and Anastasiu 2024) have traditionally been pivotal in time series forecasting due to their ability to model sequential data. Recent works have proposed novel RNN architectures and training strategies to tackle challenges like long-term dependencies and interpretability. Some models integrate autoregressive recurrent networks with probabilistic forecasting, while others combine CNNs and RNNs for enhanced multivariate prediction  (J. Li et al. 2024).

CNNs  (Wang et al. 2023) have been adapted for time series forecasting by capturing local temporal patterns and hierarchical features. Research in this area focuses on specialized CNN architectures for time series data, multi-scale convolutional operations, and hybrid models combining CNNs with other techniques for improved performance. Approaches include deep architectures with residual links and fully-connected layers  (donghao and xue 2024). However, CNNs struggle to capture long-range dependencies in time series data due to their limited receptive field size  (Wu et al. 2023). They also lack the ability to model sequential dependencies inherent in time series, as they treat each input window independently.

For multivariate and spatio-temporal forecasting tasks, GNNs  (Yi et al. 2024; Weijia Zhang et al. 2024) model dependencies between different time series as graphs. Recent studies leverage both temporal and spatial information to improve forecasting accuracy. These methods often integrate GNNs with other deep learning techniques to capture complex dependencies. However, GNNs can be computationally expensive, especially for large-scale problems such as time series and complex dependency structures  (Jin et al. 2024).

Transformer architectures  (Wu et al. 2023) have achieved significant success in time series forecasting, particularly for long-term predictions  (Vaswani et al. 2017). Recent works aim to enhance the efficiency and effectiveness of Transformers by addressing temporal dependencies, capturing multi-scale patterns, and handling non-stationarity. Some models introduce methods to capture periodic patterns at different scales, while others focus on adapting to changing data distributions  (Jia et al. 2024). However, Transformer models typically have a large number of parameters, often in the millions, leading to high computational requirements  (Liu et al. 2022).

An emerging trend involves utilizing LLMs  (T. Zhou et al. 2023; Gruver et al. 2024) for time series forecasting, taking advantage of their rich representations and transfer learning capabilities. However, due to the billions of parameters in LLMs, these approaches often require substantial computational resources  (Dooley et al. 2024), making them less suitable for resource-constrained environments like edge-cloud systems, which is the focus of our work.

Due to their simplicity and computational efficiency, Multilayer Perceptron (MLP)-based models  (Liu et al. 2024; Zeng et al. 2023; Xu, Zeng, and Xu 2024) have gained attention. Recent research explores novel MLP architectures that can capture temporal dynamics without complex attention mechanisms. Strategies include leveraging mathematical theories and frequency domain representations to model temporal dependencies  (Nie et al. 2023). However, MLP models such as SparseTSF  (Lin et al. 2024) assume a fixed periodicity in workload data, limiting their ability to adapt to changing patterns in workload data.

Our work addresses these limitations by proposing **AERO**, a lightweight adaptive edge-cloud workload forecasting model described in detail in Section .

# SYSTEM MODEL

In this section, we formalize our cloud workload prediction framework through four key components: workload representation, workload prediction model, objective function, and edge-cloud network architecture. This formalization provides the mathematical foundation for comparing different architectural approaches.

## Workload Representation

Let represent the workload vector  (H. Zhou et al. 2021) at discrete time step , where denotes the -th resource utilization feature (e.g., CPU, memory usage), is the number of features, and , with being the total number of time steps in the trace. The complete workload trace forms a multivariate time series:

## Workload Prediction Model

Our objective is to forecast future workload vectors based on historical observations. We define a prediction function , parameterized by , which represents the trainable parameters of the prediction model (e.g., weights and biases of a neural network). The function estimates the workload over a prediction horizon of length , where and :

where , , is the length of the historical context window, and denotes the sequence of past workload vectors used for prediction. The model architecture and training process are determined by hyperparameters , where defines the search space of valid configurations  (Rijn and Hutter 2018).

## Objective Function

To learn the model parameters , we minimize a loss function that measures the prediction error between the forecasted workload and the ground truth workload across all resource dimensions. We employ the Mean Squared Error (MSE) as our optimization objective, which penalizes larger prediction errors quadratically:

where denotes the Euclidean norm in and . This loss function computes the average squared distance between predicted and actual resource utilization vectors across the prediction horizon .

## Edge-cloud Network Architecture

Our simulations utilize a heterogeneous edge-cloud network comprising edge devices, fog nodes, and cloud servers  (Tuli et al. 2021). Let be the set of computational nodes. Each node possesses specific resource capacities defined as a vector , where represents CPU capacity in MHz, represents memory capacity in KB, represents storage capacity with read/write speeds in KB/s, and represents network bandwidth in Gb/s.

Let denote the set of containers, where each container is decomposed into a set of tasks for container . Each task has resource requirements and arrival time , with measurements collected at regular intervals.

Container-level workload predictions inform task scheduling across nodes to optimize resource utilization and meet Quality of Service (QoS) requirements  (Tuli et al. 2021). The scheduler employs these predictions to make proactive allocation decisions  (Farhad and Pyun 2023), aiming to minimize metrics such as energy consumption and Service Level Agreement (SLA) violations  (Raith et al. 2024). This architecture supports evaluating container workload prediction models like AERO on orchestration performance, as detailed in Section .

# AERO ARCHITECTURE

Edge-cloud environments pose unique challenges for resource orchestration due to their heterogeneity, resource constraints, and dynamic workload patterns. To address these challenges, we propose **AERO** (**A**daptive **E**dge-cloud **R**esource **O**rchestration), a lightweight forecasting model specifically tailored for adaptive workload prediction in edge-cloud settings. AERO is designed to operate efficiently on resource-constrained edge devices while maintaining the high prediction accuracy essential for effective scheduling decisions.

## Model Overview

The AERO model operates on the historical workload data , where is the context window length and is the number of features, as defined in Section . The model serves as the prediction function parameterized by , mapping historical workloads to future predictions over a horizon :

The AERO architecture consists of three main components:

1. Adaptive Period Detection Module: Identifies and adapts to the dominant period within the historical workload data by analyzing temporal patterns.
2. Adaptive Forecasting Module: Adjusts the detected period and performs forecasting using the SparseTSF model  (Lin et al. 2024). This module handles the core prediction task, leveraging the strengths of SparseTSF for efficient time series forecasting.
3. Adaptive Layer Caching: Maintains a cache of neural network layers optimized for different periodicities. When similar workload patterns recur, the model reuses these cached layers instead of recomputing them, significantly reducing computational overhead.

By integrating these components, AERO provides a scalable solution deployed on distributed edge nodes. Its lightweight design minimizes computational complexity and memory footprint, making it suitable for resource-constrained devices. With few parameters, the model enables efficient retraining and online updates directly on edge devices, allowing inexpensive adaptation to dynamic workloads. This localized approach reduces communication overhead, as each edge node updates its model independently without transmitting large data. Overall, AERO’s adaptability and efficiency ensure consistent, reliable performance across heterogeneous environments.

In the following sections, we delve into the detailed workings of each component of the AERO model. The algorithmic workflow of AERO is presented in Algorithm 1.

## Adaptive Period Detection Module

Detecting the dominant period is critical, as it directly determines the model’s configuration for optimal performance. The parameter dictates how the model segments and processes the time series data, influencing its ability to accurately capture and leverage periodic patterns. By precisely identifying , we ensure that the model aligns with the inherent periodicity of the data, effectively separating periodic components from trends and increasing forecasting accuracy.

The Adaptive Period Detection Module aims to identify the dominant period within the historical workload data sequence . First, we compute the average across the feature dimension to obtain a univariate time series :

where denotes the -th feature of the workload data sequence .

Next, we subtract the mean to focus on the periodic components, yielding the mean-centered series :

where is the mean of the averaged workload sequence.

To efficiently identify the dominant period , we begin by computing the autocorrelation function (ACF) of the mean-centered workload sequence . The autocorrelation function measures the similarity between the time series and its lagged versions, which is essential for detecting repeating patterns and periodicities in the data.

The autocorrelation function is computed using convolution  (D. Zhang et al. 2020):

where is the lag or time shift, is the time-reversed version of , denotes the convolution operation, and is the variance of . Computing the ACF via convolution allows us to leverage the Fast Fourier Transform (FFT  (Nussbaumer and Nussbaumer 1982)), reducing the computational complexity from to for large . This efficiency is important for real-time processing on resource-constrained edge devices, enabling rapid analysis of temporal dependencies without significant computational overhead.

After obtaining the ACF, we identify peaks, which are points where the autocorrelation is higher than its immediate neighbors:

These peaks correspond to potential periods in the data.

Candidate periods are selected from the peaks within the desired range (model hyperparameters):

Finally, the dominant period is assigned based on the candidate periods:

When no candidate periods are found, we default to . This choice ensures at least two complete segments within the maximum period range (a hyperparameter), providing the minimum basis for temporal comparison in the absence of periodic patterns. A larger would yield insufficient segments for comparison, while a smaller risks over-segmentation. This balanced approach maintains the model’s ability to capture potential temporal relationships even without explicit periodicity.

## Adaptive Forecasting Module

After detecting the dominant period from the historical data, the Adaptive Forecasting Module adjusts this period to ensure compatibility with the model architecture and proceeds to predict future workloads , where is the prediction horizon.

We first clamp the detected period to a valid range and make it even to satisfy the model’s requirements:

Clamping ensures that the period is within a feasible range based on the context window length , and making it even simplifies subsequent segmentation and reshaping operations.

Next, we adjust the clamped period to the closest common divisor of and :

where is the set of common divisors of and . Adjusting to ensures that both the input sequence and the prediction horizon can be evenly divided into segments of length , facilitating efficient downsampling and upsampling in the model.

After this point, the forecasting process follows the standard SparseTSF model  (Lin et al. 2024). For completeness, we outline their forecasting process below.

First, the input data is normalized by subtracting the mean vector:

where is a column vector of ones, and .

The normalized data is then transposed to align dimensions for convolution:

A one-dimensional convolution is applied with residual connection:

where the kernel size and padding .

The data is then downsampled through reshaping:

where represents the number of segments in the input sequence.

A linear transformation maps the downsampled segments to prediction segments:

where represents the number of segments in the prediction sequence.

The predictions are then upsampled through reshaping:

Finally, the predictions are denormalized:

This sequence of operations efficiently processes the time series data while maintaining its periodic structure. In our implementation, we leverage this architecture with our adaptively detected period .

## Adaptive Layer Caching Mechanism

In dynamic edge-cloud environments, workloads exhibit varying periodicities. To efficiently adapt without significant overhead, we introduce an Adaptive Layer Caching Mechanism in the AERO model. For each feasible period within a practical range , we cache pre-initialized neural network layers configured specifically for that . These layers include convolutional layers with kernel size and padding , and linear layers mapping input segments of size to output segments of size .

When the dominant period is detected (as per Section ), it may not exactly match a cached period. In such cases, we adjust to the nearest cached value using Eq. . This adjustment is acceptable because small variations in period length often correspond to similar temporal patterns in the workload data. The model’s convolutional and linear layers are designed to be robust to minor discrepancies in , effectively capturing temporal dependencies even when the period is slightly adjusted. By using the nearest cached period, we ensure that only valid, pre-initialized layers are utilized, preventing mismatches and maintaining model integrity without significant loss in prediction accuracy.

# EVALUATION

In this section, we present the evaluation of our proposed AERO model through two comprehensive scenarios. First, we conduct a performance evaluation of machine learning models, where we benchmark AERO against five state-of-the-art approaches using metrics including prediction accuracy, computational efficiency, and model complexity. Second, we perform an orchestration performance evaluation to assess how these prediction models impact fog computing environments, analyzing metrics such as energy consumption, response time, resource utilization, and SLA violations.

## Performance Evaluation of Machine Learning Models

In this section, we conduct a comprehensive machine learning evaluation to assess model performance across multiple dimensions: prediction accuracy, computational efficiency, and adaptability to dynamic workload patterns.

### Dataset

We use the Alibaba 2022 microservices trace dataset  (Luo et al. 2022), focusing on pod MS\_11349. The dataset provides samples collected at 1-minute intervals over 13 days. Each workload vector consists of features: normalized CPU utilization () and memory utilization (), aligning with the workload representation in Section . This forms a workload trace exhibiting both deterministic trends (e.g., daily patterns) and stochastic fluctuations (e.g., demand spikes), making it ideal for evaluating prediction models with different historical context windows and forecast horizons .

### Benchmarks

We compare AERO with five state-of-the-art models, each representing a distinct architecture:

### Experimental Settings

All experiments were conducted on a server with NVIDIA A100 GPU (40 GB), 30 CPU cores, 200 GiB RAM, running Ubuntu 20.04.2 LTS. Models were implemented in PyTorch, and MLFlow tracked all experiments. To ensure fair comparisons, we standardized the experimental setup. Our hyperparameter tuning method used Bayesian optimization (BoTorch  (Balandat et al. 2020), GPyTorch  (Gardner et al. 2018)) with ranges from original authors (Table ). Each model had 20 random hyperparameter evaluations, followed by 30 optimization runs (totaling 50 training runs per model), optimizing for validation loss. All parameters were logged to MLFlow  (A. Chen et al. 2020).

Per individual training run, models trained up to 100 epochs with early stopping (patience 10), optimizing validation loss. We used identical learning rate schedulers. Data were identically partitioned, and input was uniformly normalized and rescaled. Preprocessing steps ensured identical inputs. We used the same random seed for reproducibility.

By aligning hyperparameter tuning, training protocols, computational resources, data handling, and evaluation metrics, we minimized biases, ensuring fair comparisons and strengthening our conclusions.

We assess model performance using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), the two most commonly used metrics in time-series forecasting  (Mendis, Wickramasinghe, and Marasinghe 2024):

where and denote the and Euclidean norms, respectively. In addition to prediction accuracy, we evaluate practical deployment considerations including inference time, convergence speed, parameter count, and memory usage.

### Experimental Results

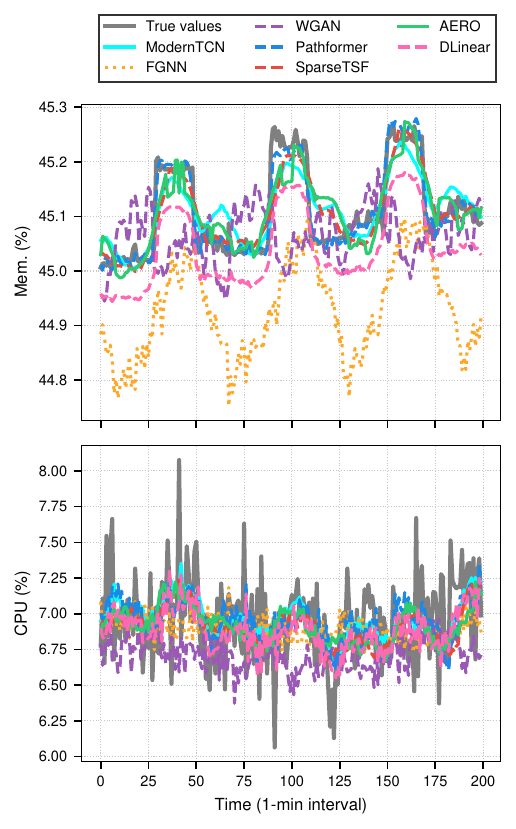
Our evaluation results are summarized in Table . We analyze the performance of each model across the defined metrics, highlighting key trade-offs relevant to cloud-native scenarios.

#### Prediction Accuracy

Accurate workload prediction is essential for efficient resource allocation and maintaining Quality of Service in cloud-native scenarios. All models demonstrate predictive capabilities, but key differences emerge (see Fig. ).

For memory utilization, Pathformer achieves the best predictive performance with an RMSE of , while ModernTCN, AERO, and SparseTSF form a competitive cluster with RMSE values of , , and , respectively. DLinear achieves moderate memory prediction performance with an RMSE of , demonstrating the trade-offs of its linear decomposition approach. This clustering suggests these architectures effectively capture the underlying patterns in memory usage, though FourierGNN and WGAN lag significantly behind with higher RMSE values, indicating potential limitations in their ability to model memory consumption patterns.

In CPU utilization prediction, we observe similar performance across all models. The RMSE values cluster tightly around , indicating that CPU patterns may be more challenging to predict precisely due to their stochastic nature.

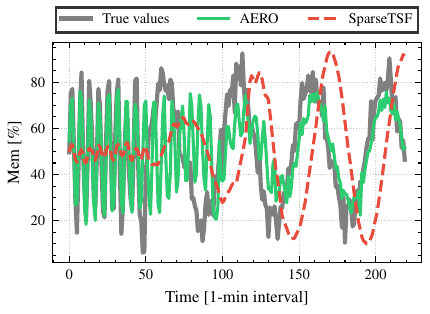
[t] 

#### Dynamic Periodicity Adaptation in AERO

To evaluate AERO’s adaptive capabilities, we compared it with SparseTSF  (Lin et al. 2024) on a synthetic dataset (5000 time steps, minute-level granularity) featuring dynamic periodic shifts in CPU and memory utilization. The trace incorporates periods of 6, 24, 48, and 96 time steps to simulate varying workload phases. While SparseTSF operates under a fixed periodicity assumption, AERO dynamically detects and adapts to the dominant periodicity.

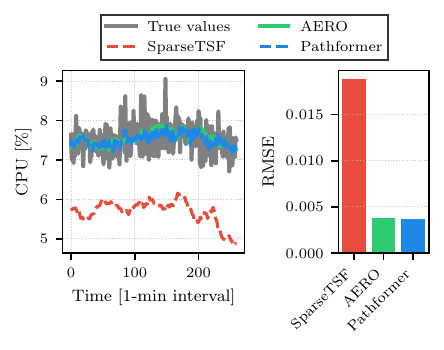
As shown in Fig. , SparseTSF fails to capture both high- and low-frequency patterns due to its static periodicity assumption. In contrast, AERO successfully adapts to shifting periodicities, achieving superior performance (RMSE: 0.180, MAE: 0.122) compared to SparseTSF (RMSE: 0.315, MAE: 0.273).

AERO requires approximately two periods () to adjust to periodicity changes, as observed in Fig. . This adaptation window stems from AERO’s autocorrelation function (ACF) mechanism (Equation , Section ), which needs sufficient data to detect new periodic patterns through Equation . As more data from the period is collected, the ACF at lag increases. After about two full cycles of , the repeated patterns are sufficient for the ACF to detect as the new dominant period. The Adaptive Forecasting Module then segments the data based on the detected period . Having at least two complete periods ensures that the segmentation captures the new periodic pattern effectively, improving prediction accuracy.



#### Robust Performance on Stochastic Cloud Workloads

Fig.  illustrates AERO’s ability to handle stochastic cloud workloads. The ground truth shows temporal variance and abrupt transitions, challenging for models relying on periodic patterns. Without clear periodicity, AERO’s period detection defaults to minimal periods, dynamically adjusting convolution kernel size and padding to leverage recent patterns. With only 559 parameters, AERO achieves predictive accuracy comparable to Pathformer (2.6 million parameters), evidenced by similar RMSE scores (0.0035) in zero-shot prediction on unseen workloads. This suggests higher model complexity does not necessarily improve predictions for stochastic events; AERO’s adaptive forecasting efficiently processes time series data while maintaining local structure. In contrast, SparseTSF, assuming fixed periodicity, yields an RMSE about five times higher, highlighting reduced effectiveness in capturing stochastic behavior. These findings demonstrate AERO’s suitability for real-world deployment, effectively modeling non-periodic dynamics.

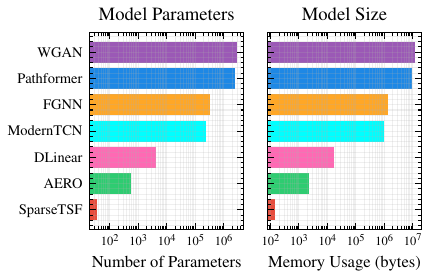
[t] 

#### Model Complexity and Size

Fig.  presents a logarithmic comparison of model complexity, highlighting the lightweight nature of AERO relative to more complex architectures.

Resource efficiency is critical in cloud-native deployments, particularly at the edge  (Waleffe et al. 2023). SparseTSF stands out with only 35 parameters and minimal memory usage (140 bytes), ideal for resource-constrained devices. AERO has 599 parameters and a memory footprint of 2,236 bytes, balancing lightweight design with enhanced functionality. DLinear occupies a middle ground with 4128 parameters, offering a practical balance between model complexity and predictive power. While larger than SparseTSF, AERO maintains a sub-1K parameter count while offering adaptability to varying periodicities.

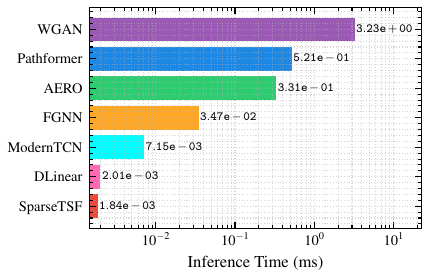
ModernTCN and FourierGNN offer a balance between complexity and performance, suitable for environments with moderate resources. Pathformer and WGAN, with millions of parameters and higher memory requirements, are more appropriate for powerful edge nodes or cloud servers.

[t] 

#### Inference Time

Fig.  shows the inference times for all models, highlighting the trade-off between complexity and latency.

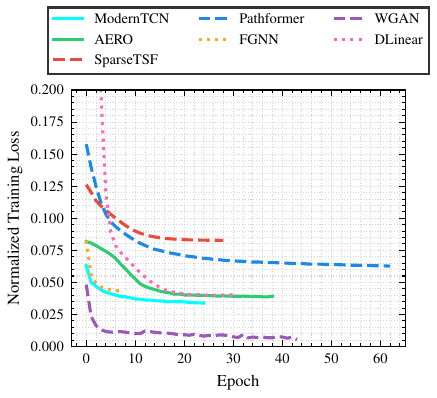
Inference time was measured per batch over 1000 batches and averaged. Low inference latency is critical for real-time decision-making in cloud-native environments. SparseTSF demonstrates the fastest inference time (s), while AERO has an inference time of  ms. DLinear achieves a competitive inference time of s, positioning it favorably among the more efficient models. Since both models operate well below the 50 ms latency threshold required for real-time applications  (Schulz et al. 2017), their performance difference is not significant for practical deployments. AERO balances this latency increase with adaptive capabilities (Fig. ), making it suitable for scenarios where adaptability to varying periodicities is essential.

[t] 

#### Convergence Behavior

Fig.  illustrates the normalized training loss convergence across models. AERO demonstrates stable convergence, with a slight delay compared to the fastest models. It is important to note that models with very low training loss may be susceptible to overfitting.

Fast convergence is essential for models requiring frequent retraining in dynamic cloud-native workloads  (Gudepu et al. 2023). SparseTSF, ModernTCN, and FourierGNN converge rapidly within 10 epochs, facilitating quick adaptation to new data. DLinear demonstrates stable convergence in 17 epochs, closely matching AERO’s convergence characteristics while maintaining consistent training stability. AERO converges in 16 epochs, reflecting the additional complexity from its adaptive period detection module. This slight increase in training time is offset by improved adaptability to varying workload patterns. WGAN requires 43 epochs to converge and exhibits higher variability during training, likely due to instability from its adversarial training process.

[t] 

#### Observations and Discussion

Our experimental results reveal key performance characteristics across models. For memory prediction, Pathformer achieves the highest accuracy (RMSE: ), while ModernTCN, AERO, and SparseTSF show competitive performance (RMSE: -). CPU prediction yields similar performance across all models (RMSE approximately ), suggesting inherent challenges in predicting stochastic CPU patterns.

AERO extends SparseTSF’s capabilities by incorporating adaptive period detection while maintaining computational efficiency (Section ). The models span a wide resource spectrum: SparseTSF is extremely lightweight (35 parameters), Pathformer is resource-intensive (2M+ parameters), and AERO offers a balanced middle ground with fewer than 1,000 parameters.

Based on these findings, we select Pathformer, SparseTSF, and AERO for further simulation analysis, as they represent distinct approaches to the prediction problem and enable comprehensive evaluation of accuracy-efficiency trade-offs in cloud workload prediction.

## Simulations for Orchestration Performance of Workload Predictors

In this section, we conduct simulations to evaluate the impact of workload prediction on container orchestration performance in fog computing environments. Using the COSCO framework  (Tuli et al. 2021), we simulate a heterogeneous environment combining both edge and cloud resources to realistically assess our approach. We begin by detailing our experimental configuration, outlining the specifics of the simulated environment and the scheduling setup. Next, we describe how workload prediction models are integrated into the simulator, highlighting the methods for incorporating predictions into the scheduling decisions. Following this, we explain the dual-criteria orchestration objective function that balances energy efficiency and response time, forming the basis of our performance evaluation. We then analyze the BitBrains workload dataset used for model training, discussing its characteristics and why it is suitable for our simulations. Finally, we assess the impact of workload predictions on orchestration performance by comparing key metrics across our selected predictive models.

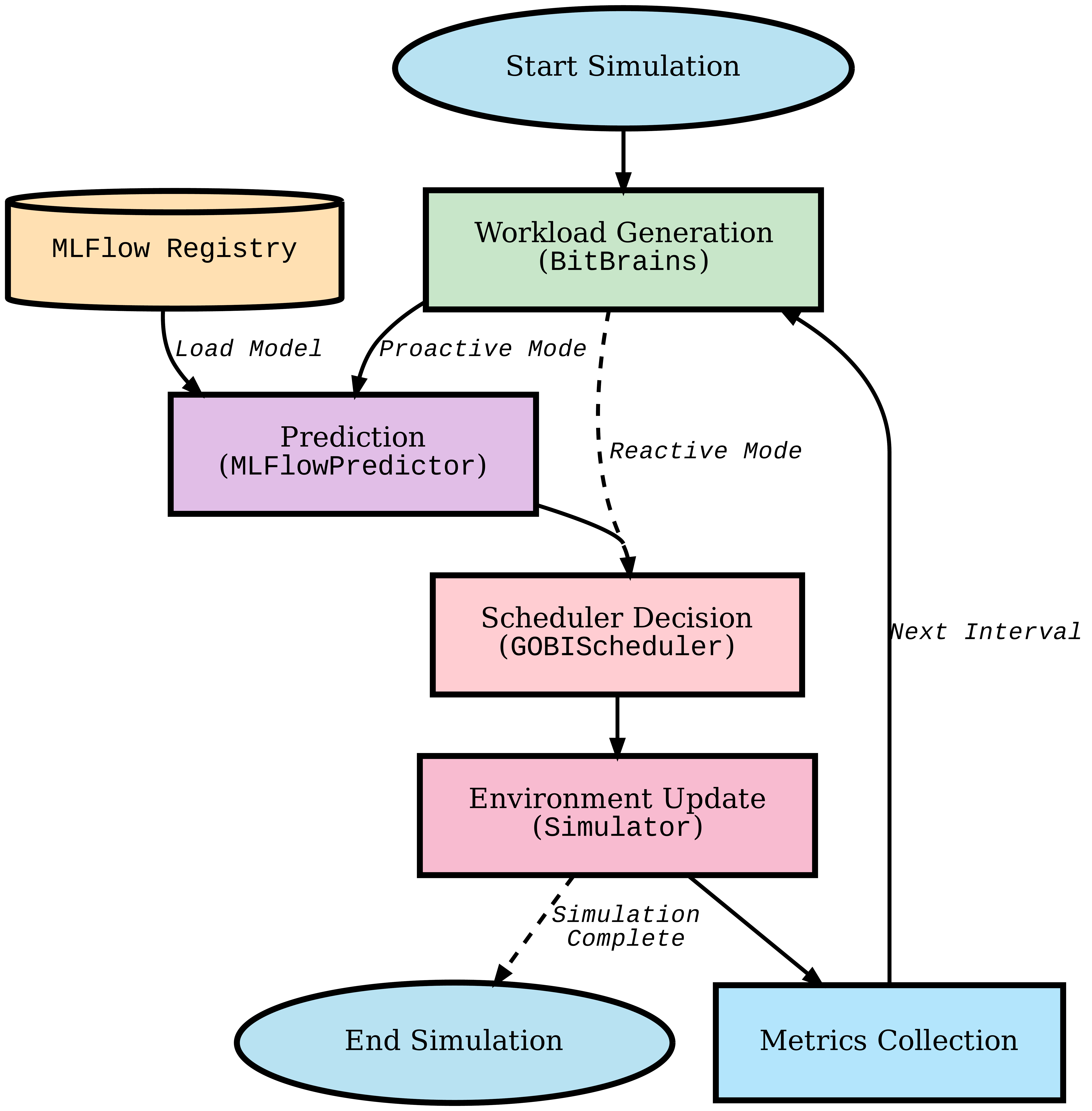
### Experimental Configuration

We evaluate our scheduling approach using the COSCO framework  (Tuli et al. 2021), which simulates realistic edge-cloud infrastructures. Their deep learning GOBIScheduler orchestrates containers across the distributed infrastructure , consisting of heterogeneous computational nodes with a total power capacity of and inter-node network bandwidth of . The simulation executes discrete time steps at intervals, with containers and new container introductions per interval.

We use the AzureFog configuration with 60% edge nodes and 40% cloud nodes. The nodes consist of three VM types: B2s VMs (40% of nodes) with 4,029 Million Instructions Per Second (MIPS), 4GB Random Access Memory (RAM), and 32GB storage; B4ms VMs (40%) with 4,029 MIPS, 16GB RAM, and 32GB storage; and B8ms VMs (20%) with 16,111 MIPS, 32GB RAM, and 32GB storage.

Edge nodes have a network latency of 3 ms, while cloud nodes operate at 76 ms. All nodes are equipped with 5Gbps network bandwidth for both uplink and downlink. Each node has varying capacities for CPU, memory, disk, and network resources, as defined in Section . Each container comprises tasks , where each task arrives at time and requires resources .

### Integration of Workload Prediction Models



We have evaluated AERO in a simulated environment by extending the COSCO framework  (Tuli et al. 2021) with a prediction interface integrated via MLFlow  (A. Chen et al. 2020), as shown in Fig. 7. The simulator operates in two modes:

* **Proactive Mode**: Uses pre-trained models from the MLFlow Registry to forecast future resource demands, feeding predictions to the GOBIScheduler for proactive resource allocation.
* **Reactive Mode**: Runs the GOBIScheduler without workload predictions, making decisions based solely on the current network state. The scheduler checks resource usage thresholds and reallocates containers upon violations, reacting to immediate bottlenecks without anticipating workload spikes.

By keeping the scheduler identical across experiments and varying only the prediction inputs, we isolate the impact of workload prediction models on orchestration performance. For detailed information on the scheduler’s algorithm and implementation, please refer to  (Tuli et al. 2021). We explain the schedulers’ objective shortly in the next section.

### Dual-Criteria Orchestration Objective Function

The GOBIScheduler minimizes a normalized dual-objective function over each interval that balances energy efficiency and response time  (Tuli et al. 2021), two of the most important metrics for edge-cloud computing  (McChesney et al. 2019):

where controls the trade-off between energy consumption and response time, and represents the system state at time . The normalized energy consumption measures the total power used across all nodes relative to their maximum capacity:

where is the instantaneous power consumption of node at time , is its maximum power capacity, is the set of active containers, and is the interval duration. The normalized response time quantifies task completion efficiency across the container set :

where represents the completion time for task .

### BitBrains Workload Dataset Analysis and Model Training

We use the BitBrains Random trace  (Dang-Quang and Yoo 2021) for our simulation evaluations. The dataset consists of traces from 500 virtual machines connected to Storage Area Network (SAN) and Network Attached Storage (NAS) devices. One of each 500 files provide samples collected at 300 ms intervals. Each workload vector consists of features, as described in Table , aligning with the workload representation in Section .

Table 2: Key resource usage features being predicted by the models

| **Feature** | **Description** |
| --- | --- |
|  | Raw CPU computational power |
|  | CPU utilization percentage |
|  | RAM consumption |
|  | Disk read throughput |
|  | Disk write throughput |
|  | Incoming network throughput |
|  | Outgoing network throughput |

Following the methodology in Section , we trained AERO and compared its performance with Pathformer and SparseTSF on the BitBrains dataset. Fig.  shows the RMSE comparisons across various resource metrics. Consistent with previous findings, AERO demonstrates superior performance across most resource dimensions.

Notably, AERO outperforms the baselines in five out of seven metrics, with significant improvements in CPU utilization prediction (RMSE of 2.30 MHz vs. 5.96 MHz for Pathformer) and network receive throughput (0.12 KB/s vs. 0.42 KB/s). This advantage stems from AERO’s ability to dynamically adapt to workload periodicities (Section ), whereas SparseTSF’s fixed periodicity assumption limits its adaptability. Our experiments demonstrate AERO’s ability to handle mid-range dimensional workloads effectively . Despite the increased complexity of predicting multiple resource usage metrics simultaneously.

While SparseTSF slightly outperforms AERO in memory usage prediction (RMSE of 25.54 MB compared to 26.09 MB for AERO), AERO’s consistent performance across multiple metrics, along with its lightweight architecture (Section ), underscores its suitability for real-time cloud orchestration in highly dynamic environments.

### Impact of Workload Predictions on Orchestration Performance

To assess how workload predictions influence the orchestration objectives defined in Section , we compared our AERO model with other predictive schedulers (Pathformer, SparseTSF) and a Reactive baseline that does not use predictions. As shown in Fig. , we evaluated key performance metrics—energy consumption , response time , resource utilization, and Service Level Agreement (SLA) violations—based on the dual-objective function in Equation .

#### Energy Consumption and CPU Utilization

Fig.  compares the energy consumption and CPU utilization across different models. The Reactive approach consumes significantly more energy, with a mean of  J, compared to  J for both AERO and Pathformer. This higher energy usage is directly related to increased CPU utilization, as shown in Fig. .

Without workload predictions, the Reactive model cannot anticipate resource demands, leading to delayed scaling. As workloads spike unexpectedly, CPU utilization surges, causing higher energy consumption according to Equation . Predictive models like AERO forecast future demands, allowing the scheduler to adjust resources proactively. This foresight results in smoother CPU usage and reduced energy consumption, optimizing the objective .

#### Response Time and SLA Violations

Timely resource provisioning also impacts response time (Equation ) and SLA violations. The Reactive model experiences higher response times, with a mean of  s, compared to  s for AERO and  s for Pathformer (Fig. ). Late adjustments to workload changes mean tasks wait longer, increasing and negatively affecting .

Additionally, the Reactive approach shows a higher SLA violation rate (Fig. ) at , whereas AERO and Pathformer maintain low rates of and , respectively. By predicting workload spikes, AERO ensures resources are available when needed, minimizing response times and SLA violations, thus improving overall orchestration performance.

#### Task Migrations and Resource Utilization

As shown in Fig. , the Reactive model performs the fewest task migrations (2,026), while the predictive models exhibit more migrations: SparseTSF (5,309), AERO (7,747), and Pathformer (8,480). This increase is due to proactive workload predictions enabling the scheduler to reallocate tasks in anticipation of resource demand spikes.

Such migrations enable the system to adapt to dynamic workload conditions, ensuring compliance with SLAs and improving overall performance. By proactively redistributing tasks, the scheduler prevents resource overloads and balances utilization across the network. Moreover, in scenarios where workloads are low across different servers, migrations facilitate server consolidation by moving tasks to shared machines, resulting in significant energy savings. As indicated in Fig. , predictive models maintain steadier CPU utilization compared to the Reactive model, which suffers from resource hotspots due to its inability to anticipate workload spikes.

#### Overall Implications

Our findings indicate that incorporating workload predictions significantly improves the dual-objective function (Equation ). Predictive models like AERO enable proactive resource management, reducing both energy consumption and response time. In dynamic fog computing environments, predictive orchestration is crucial for optimizing performance metrics.

While accurate predictions are beneficial, the adaptability and efficiency of the orchestration model are more critical for improving . AERO balances prediction accuracy with computational efficiency, making it suitable for real-time deployments without excessive overhead.

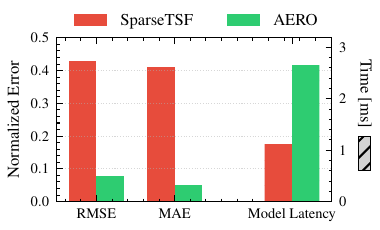
## Deployment in a Real Edge-Cloud Environment

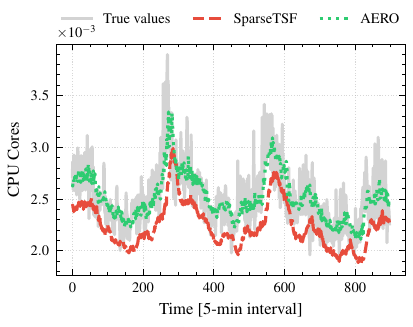
We have deployed AERO and SparseTSF as microservices within our edge-cloud testbed, featuring an Intel i9-9900K CPU, 46 GB RAM, and an NVIDIA RTX 2060 GPU running Ubuntu 24.04.1 LTS. Docker Compose was used to define and manage the multi-container deployment, providing a realistic environment.

Using this infrastructure, we maintained the same experimental setup and hyperparameter optimization as described in Section . Both models were trained and tested on the Google Cluster Workload Traces  (Tirmazi et al. 2020). The dataset has four features per 5-minute interval: average CPU cores used, average memory usage, maximum CPU cores used, and maximum memory usage. We have selected a random trace from the dataset, containing data points for training purposes. On a hold-out test set derived from the training data, both models exhibited similar performance, achieving RMSE values of 0.020 and MAE of 0.007, indicating comparable predictive capabilities under controlled conditions.

For live deployment, another random trace generated the workload for a real microservice application. In this setting, AERO significantly outperformed SparseTSF, achieving a normalized RMSE of 0.079 and MAE of 0.051 (Fig. ), compared to SparseTSF’s RMSE of 0.430 and MAE of 0.411. Fig.  shows that AERO closely tracks true usage patterns, while SparseTSF deviates considerably. These results demonstrate AERO’s superior adaptability to dynamic workloads in a live setting, consistent with Section V.C The comparable performance on the hold-out test set contrasts with the significant difference in live deployment, highlighting the importance of evaluating models under zero-shot and real-world conditions.

The mean inference time for AERO was 2.653 ms, versus 1.117 ms for SparseTSF. Despite longer inference time due to its adaptive features, AERO’s latency remains well below the 50 ms threshold for real-time applications  (Schulz et al. 2017). The increased computational overhead is justified by significant improvements in prediction accuracy. As demonstrated in simulations (Section V.B), better predictions lead to reductions of 15% in average application response time and 12% in total energy consumption. These results confirm AERO’s effectiveness in real-world deployments, enhancing resource allocation efficiency.



[t] 

# CONCLUSION

We introduced **AERO**, a lightweight and adaptive forecasting model for resource management in edge-cloud networks. AERO dynamically detects and adjusts to dominant periodicities in workload data, overcoming the limitations of fixed-periodicity models. With its minimal parameter count, it is suitable for resource-constrained environments.

Evaluations on real-world and live deployments datasets demonstrate that AERO achieves comparable prediction accuracy and operational performance to models with highly advanced architectures and millions of parameters, such as Pathformer, while significantly reducing complexity and computational overhead. Simulations using the extended COSCO framework show that integrating AERO enhances energy efficiency, reduces response times, and minimizes SLA violations compared to reactive approaches. Our live deployment experiments further validate these findings, demonstrating that AERO consistently delivers superior performance.

These findings underscore the importance of adaptive prediction models in efficiently managing fluctuating workloads in dynamic cloud-edge environments. Future work includes integrating AERO into real-time orchestration systems and examining its scalability in larger deployments.

# BIOGRAPHIES

# References

Arbat, Shivani, Vinodh Kumaran Jayakumar, Jaewoo Lee, Wei Wang, and In Kee Kim. 2022. “Wasserstein Adversarial Transformer for Cloud Workload Prediction.” In *Proceedings of the AAAI Conference on Artificial Intelligence*, 36:12433–39. 11.

Balandat, Maximilian, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. 2020. “BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization.” *Advances in Neural Information Processing Systems* 33: 21524–38.

Benidis, Konstantinos, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, et al. 2022. “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey.” *ACM Computing Surveys* 55 (6): 1–36.

Chen, Andrew, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, et al. 2020. “Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle.” In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. Deem ’20. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3399579.3399867>.

Chen, Peng, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. 2024. “Pathformer: Multi-Scale Transformers with Adaptive Pathways for Time Series Forecasting.” In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=lJkOCMP2aW>.

Collet, Alan, Antonio Bazco-Nogueras, Albert Banchs, and Marco Fiore. 2023. “Automanager: A Meta-Learning Model for Network Management from Intertwined Forecasts.” In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 1–10. <https://doi.org/10.1109/infocom53939.2023.10229064>.

Dang-Quang, Nhat-Minh, and Myungsik Yoo. 2021. “Multivariate Deep Learning Model for Workload Prediction in Cloud Computing.” In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 858–62. <https://doi.org/10.1109/ictc52510.2021.9620931>.

donghao, Luo, and wang xue. 2024. “ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis.” In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=vpJMJerXHU>.

Dooley, Samuel, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V Naidu, and Colin White. 2024. “Forecastpfn: Synthetically-Trained Zero-Shot Forecasting.” *Advances in Neural Information Processing Systems* 36.

Ekambaram, Vijay, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. “Tsmixer: Lightweight Mlp-Mixer Model for Multivariate Time Series Forecasting.” In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 459–69.

Fan, Xiaochen, Chaocan Xiang, Liangyi Gong, Xiangjian He, Chao Chen, and Xiang Huang. 2019. “UrbanEdge: Deep Learning Empowered Edge Computing for Urban IoT Time Series Prediction.” In *Proceedings of the ACM Turing Celebration Conference-China*, 1–6.

Farhad, Arshad, and Jae-Young Pyun. 2023. “AI-ERA: Artificial Intelligence-Empowered Resource Allocation for LoRa-Enabled IoT Applications.” *IEEE Transactions on Industrial Informatics* 19 (12): 11640–52.

Gardner, Jacob, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. 2018. “Gpytorch: Blackbox Matrix-Matrix Gaussian Process Inference with Gpu Acceleration.” *Advances in Neural Information Processing Systems* 31.

Gort, Berend J. D., Godfrey M. Kibalya, Maria A. Serrano, and Angelos Antonopoulos. 2025. “Forecasting Trends in Cloud-Edge Computing: Unleashing the Power of Attention Mechanisms.” *IEEE Communications Magazine* 63 (1): 108–14. <https://doi.org/10.1109/MCOM.001.2300583>.

Gruver, Nate, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2024. “Large Language Models Are Zero-Shot Time Series Forecasterss.” *Advances in Neural Information Processing Systems* 36.

Gudepu, Venkateswarlu, Venkatarami Reddy Chintapalli, Piero Castoldi, Luca Valcarenghi, Bheemarjuna Reddy Tamma, and Koteswararao Kondepu. 2023. “Adaptive Retraining of AI/ML Model for Beyond 5G Networks: A Predictive Approach.” In *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, 282–86. IEEE.

Guo, Jing et al. 2019. “Who Limits the Resource Efficiency of My Datacenter: An Analysis of Alibaba Datacenter Traces.” In *2019 IEEE/ACM 27th International Symposium on Quality of Service (Iwqos)*, 1–10. <https://doi.org/10.1145/3326285.3329074>.

Hochreiter, Sepp. 1998. “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions.” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (02): 107–16.

Iqbal, W., J. L. Berral, A. Erradi, and D. Carrera. 2019. “Adaptive Prediction Models for Data Center Resources Utilization Estimation.” *IEEE Transactions on Network and Service Management* 16 (4): 1681–93.

Jhin, Sheo Yon, Seojin Kim, and Noseong Park. 2024. “Addressing Prediction Delays in Time Series Forecasting: A Continuous GRU Approach with Derivative Regularization.” In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 33:1234–45. Kdd ’24. ACM. <https://doi.org/10.1145/3637528.3671969>.

Jia, Yuxin, Youfang Lin, Xinyan Hao, Yan Lin, Shengnan Guo, and Huaiyu Wan. 2024. “Witran: Water-Wave Information Transmission and Recurrent Acceleration Network for Long-Range Time Series Forecasting.” *Advances in Neural Information Processing Systems* 36.

Jin, Ming, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. 2024. “A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kardani-Moghaddam, S., R. Buyya, and K. Ramamohanarao. 2020. “ADRL: A Hybrid Anomaly-Aware Deep Reinforcement Learning-Based Resource Scaling in Clouds.” *IEEE Transactions on Parallel and Distributed Systems* 32 (3): 514–26.

Karim, M. E., M. M. S. Maswood, S. Das, and A. G. Alharbi. 2021. “Bhyprec: A Novel Bi-LSTM Based Hybrid Recurrent Neural Network Model to Predict the CPU Workload of Cloud Virtual Machine.” *IEEE Access* 9: 131476–95.

Kong, Xiangjie, Yuhan Wu, Hui Wang, and Feng Xia. 2022. “Edge Computing for Internet of Everything: A Survey.” *IEEE Internet of Things Journal* 9 (23): 23472–85.

Kumar, J., and A. K. Singh. 2018. “Workload Prediction in Cloud Using Artificial Neural Network and Adaptive Differential Evolution.” *Future Generation Computer Systems* 81: 41–52.

Kumar, J., A. K. Singh, and R. Buyya. 2021. “Self Directed Learning Based Workload Forecasting Model for Cloud Resource Management.” *Information Sciences* 543: 345–66.

Lai, Guokun, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. “Modeling Long-and Short-Term Temporal Patterns with Deep Neural Networks.” In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Li, Jialun, Jieqian Yao, Danyang Xiao, Diying Yang, and Weigang Wu. 2024. “EvoGWP: Predicting Long-Term Changes in Cloud Workloads Using Deep Graph-Evolution Learning.” *IEEE Transactions on Parallel and Distributed Systems*.

Li, Yanhong, Jack Xu, and David Anastasiu. 2024. “Learning from Polar Representation: An Extreme-Adaptive Model for Long-Term Time Series Forecasting.” In *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:171–79. 1.

Lin, Shengsheng, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. 2024. “SparseTSF: Modeling Long-Term Time Series Forecasting with 1k Parameters.” In *International Conference on Machine Learning*.

Liu, Yong, Chenyu Li, Jianmin Wang, and Mingsheng Long. 2024. “Koopa: Learning Non-Stationary Time Series Dynamics with Koopman Predictors.” *Advances in Neural Information Processing Systems* 36.

Liu, Yong, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. “Non-Stationary Transformers: Exploring the Stationarity in Time Series Forecasting.” *Advances in Neural Information Processing Systems* 35: 9881–93.

Lu, C. et al. 2017. “Imbalance in the Cloud: An Analysis on Alibaba Cluster Trace.” In *2017 IEEE International Conference on Big Data (Big Data)*, 2884–92.

Luo, Shutian, Huanle Xu, Kejiang Ye, Guoyao Xu, Liping Zhang, Guodong Yang, and Chengzhong Xu. 2022. “The Power of Prediction: Microservice Auto Scaling via Workload Learning.” In *Proceedings of the ACM Symposium on Cloud Computing*.

McChesney, Jonathan, Nan Wang, Ashish Tanwer, Eyal De Lara, and Blesson Varghese. 2019. “Defog: Fog Computing Benchmarks.” In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 47–58.

Mendis, Kasun, Manjusri Wickramasinghe, and Pasindu Marasinghe. 2024. “Multivariate Time Series Forecasting: A Review.” In *Proceedings of the 2024 2nd Asia Conference on Computer Vision, Image Processing and Pattern Recognition*. CVIPPR ’24. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3663976.3664241>.

Meuser, Tobias, Lauri Lovén, Monowar Bhuyan, Shishir G Patil, Schahram Dustdar, Atakan Aral, Suzan Bayhan, et al. 2024. “Revisiting Edge Ai: Opportunities and Challenges.” *IEEE Internet Computing* 28 (4): 49–59.

Nie, Yuqi, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. “A Time Series Is Worth 64 Words: Long-Term Forecasting with Transformers.” In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=Jbdc0vTOcol>.

Nussbaumer, Henri J, and Henri J Nussbaumer. 1982. *The Fast Fourier Transform*. Springer.

Ouyang, Tao, Kongyange Zhao, Xiaoxi Zhang, Zhi Zhou, and Xu Chen. 2023. “Dynamic Edge-Centric Resource Provisioning for Online and Offline Services Co-Location.” In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 1–10. IEEE.

Raith, Philipp, Gourav Rattihalli, Aditya Dhakal, Sai Rahul Chalamalasetti, Dejan Milojicic, Eitan Frachtenberg, Stefan Nastic, and Schahram Dustdar. 2024. “Opportunistic Energy-Aware Scheduling for Container Orchestration Platforms Using Graph Neural Networks.” In *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 299–306. IEEE.

Rijn, Jan N. van, and Frank Hutter. 2018. “Hyperparameter Importance Across Datasets.” In *Proceedings of the 24th ACM Sigkdd International Conference on Knowledge Discovery & Data Mining*, 2367–76. Kdd ’18. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3219819.3220058>.

Saxena, D., J. Kumar, A. K. Singh, and S. Schmid. 2023. “Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud.” *IEEE Transactions on Parallel and Distributed Systems* 34 (4): 1313–30.

Schulz, Philipp, Maximilian Matthe, Henrik Klessig, Meryem Simsek, Gerhard Fettweis, Junaid Ansari, Shehzad Ali Ashraf, et al. 2017. “Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture.” *IEEE Communications Magazine* 55 (2): 70–78.

Shang, Xiaojun, Yingling Mao, Yu Liu, Yaodong Huang, Zhenhua Liu, and Yuanyuan Yang. 2023. “Online Container Scheduling for Data-Intensive Applications in Serverless Edge Computing.” In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 1–10. <https://doi.org/10.1109/infocom53939.2023.10229034>.

Singh, A. K., D. Saxena, J. Kumar, and V. Gupta. 2021. “A Quantum Approach Towards the Adaptive Prediction of Cloud Workloads.” *IEEE Transactions on Parallel and Distributed Systems* 32 (11): 2893–2907.

Singh, N., and S. Rao. 2014. “Ensemble Learning for Large-Scale Workload Prediction.” *IEEE Transactions on Emerging Topics in Computing* 2 (2): 149–65.

Tang, X. 2019. “Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network.” *IEEE Access* 7: 40525–33.

Tirmazi, Muhammad et al. 2020. “Borg: the next generation.” In *Proceedings of the Fifteenth European Conference on Computer Systems*, 1–14.

Tuli, Shreshth, Shivananda R Poojara, Satish N Srirama, Giuliano Casale, and Nicholas R Jennings. 2021. “COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments.” *IEEE Transactions on Parallel and Distributed Systems* 33 (1): 101–16.

Vaswani, Ashish et al. 2017. “Attention Is All You Need.” *Advances in Neural Information Processing Systems* 30.

Waleffe, Roger, Jason Mohoney, Theodoros Rekatsinas, and Shivaram Venkataraman. 2023. “MariusGNN: Resource-Efficient Out-of-Core Training of Graph Neural Networks.” In *Proceedings of the Eighteenth European Conference on Computer Systems*, 144–61.

Wang, Huiqiang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. “MICN: Multi-Scale Local and Global Context Modeling for Long-Term Series Forecasting.” In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=zt53IDUR1U>.

Wu, Haixu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. “TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis.” In *International Conference on Learning Representations*.

Xu, Zhijian, Ailing Zeng, and Qiang Xu. 2024. “FITS: Modeling Time Series with $10k$ Parameters.” In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=bWcnvZ3qMb>.

Yi, Kun, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. 2024. “FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective.” *Advances in Neural Information Processing Systems* 36.

Zeng, Ailing, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. “Are Transformers Effective for Time Series Forecasting?” In *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:11121–28. 9.

Zhang, Duona, Wenrui Ding, Hongyu Wang, and Baochang Zhang. 2020. “Autocorrelation Convolution Networks Based on Deep Learning for Automatic Modulation Classification.” In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 1561–65. IEEE.

Zhang, Q., L. T. Yang, Z. Yan, Z. Chen, and P. Li. 2018. “An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics.” *IEEE Transactions on Industrial Informatics* 14 (7): 3170–78.

Zhang, W., P. Duan, L. T. Yang, F. Xia, Z. Li, Q. Lu, W. Gong, and S. Yang. 2017. “Resource Requests Prediction in the Cloud Computing Environment with a Deep Belief Network.” *Software: Practice and Experience* 47 (3): 473–88.

Zhang, Weijia, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. 2024. “Irregular Multivariate Time Series Forecasting: A Transformable Patching Graph Neural Networks Approach.” In *Forty-First International Conference on Machine Learning*. <https://openreview.net/forum?id=UZlMXUGI6e>.

Zhou, Haoyi et al. 2021. “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting.” In *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:11106–15. 12.

Zhou, Tian, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. “One Fits All: Power General Time Series Analysis by Pretrained LM.” *Advances in Neural Information Processing Systems* 36: 43322–55.