MSCNet: Multi-Scale Network With Convolutions for Long-Term Cloud Workload Prediction

Feiyu Zhao, Weiwei Lin, *Senior Member, IEEE*, Shengsheng Lin, Shaomin Tang, and Keqin Li, *Fellow, IEEE*

***Abstract*—Accurate workload prediction is crucial for resource allocation and management in large-scale cloud data centers. While many approaches have been proposed, most existing methods are based on Recurrent Neural Networks (RNNs) or their variants, focusing on short-term cloud workload prediction without considering or identifying the long-term changes and different periodic patterns of cloud workloads. Due to variations in user demands or workload dynamics, cloud workloads that appear stable in the short term often exhibit distinct patterns in the long term. This can lead to a significant decline in prediction accuracy for existing methods when applied to long-term cloud workload forecasting. To address these challenges and overcome the limitations of current approaches, we propose a Multi-Scale Network with Convolutions (MSCNet) for accurate long-term cloud workload prediction. MSCNet employs multi-scale modeling of the original cloud workload to effectively extract multi-scale features and different periodic patterns, learning the long-term dependencies among the cloud workload. Our core component, the Multi-Scale Block, combines the Multi-Scale Patch Block, Transformer Encoder, and Multi-Scale Convolutions Block for comprehensive multi-scale learning. This enables MSCNet to adaptively learn both short-term and long-term features and patterns of cloud workloads, resulting in accurate long-term cloud workload predictions. Extensive experiments are conducted using real-world cloud workload data from Alibaba, Google, and Azure to validate the effectiveness of MSCNet. The experimental results demonstrate that MSCNet achieves accurate long-term cloud workload prediction with a computational complexity of $O(L^2d)$, outperforming existing state-of-the-art methods.**

***Index Terms*—Cloud computing, workload prediction, multi-scale, long-term, resource allocation.**

Received 27 June 2024; revised 5 November 2024; accepted 18 January 2025. Date of publication 30 January 2025; date of current version 10 April 2025. This work was supported in part by Guangdong Major Project of Basic and Applied Basic Research under Grant 2019B030302002, in part by National Natural Science Foundation of China Grant 62072187, in part by Guangzhou Development Zone Science and Technology Project under Grant 2023GH02, and in part by the Major Key Project of PCL, China under Grant PCL2023A09. *(Corresponding author: Weiwei Lin.)*

Feiyu Zhao is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: zhaofy001@foxmail.com).

Weiwei Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Pengcheng Laboratory, Shenzhen 518066, China (e-mail: linww@scut.edu.cn).

Shengsheng Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: linss2000@foxmail.com).

Shaomin Tang is with the School of Electronics and Information Engineering, South China Normal University, Guangzhou 510631, China (e-mail: tangshaomin@scnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TSC.2025.3536313

I. Introduction

CLOUD computing has emerged as a prominent form of distributed computing, providing dynamic resources such as processing, storage, and bandwidth-on-demand for customers. It utilizes virtualization techniques to ensure scalability and reliability, meeting the Quality of Service (QoS) stipulations outlined in Service Level Agreements (SLAs) [1]. These resources, including CPU capacity, memory, and I/O, are essential for deploying any application in the cloud, and they are provided as services by Cloud Service Providers (CSPs). The flexibility and affordability of cloud services have made them increasingly popular for supporting a diverse array of applications, including Big Data processing, machine learning, web hosting, and more [2].

One of the key challenges in cloud computing is efficiently managing resources to meet SLAs while minimizing resource wastage. Over-provisioning resources is a common strategy to ensure SLA compliance, but it leads to significant energy and resource wastage. Conversely, under-provisioning results in SLA violations and customer dissatisfaction [3]. Accurate cloud workload prediction methods forecast future workloads based on previous traces, aiding CSPs in capacity planning. Proactive resource management, which predicts future workload and adjusts resource allocation accordingly, can prevent SLA violations and reduce energy consumption [1]. Therefore, CSPs require the ability to promptly identify resource provisioning strategies, ensuring compliance with SLA requirements while optimizing resource utilization [4]. To achieve these goals, cloud computing demands precise and efficient workload prediction methods. By accurately predicting future workloads, CSPs can proactively configure and allocate resources in advance, optimizing resource provisioning for greater efficiency and rationality [5].

Due to its significance, considerable efforts have been directed toward workload prediction. These methods can mainly be classified into two categories: regression-based methods [6], [7] [8] and machine learning-based methods. Regression-based methods, such as ARIMA, assume that time series are stationary, whereas real-world cloud workload series are non-stationary. Additionally, ARIMA struggles to learn long-term dependencies between cloud workloads and is only suitable for short-term prediction [9]. In contrast, most existing machine learning-based methods use RNN or RNN-based models to learn the dependencies between cloud workload series or different workload

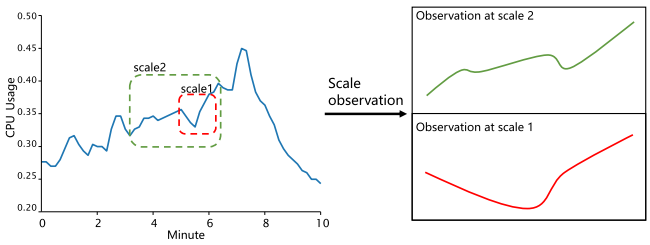


Fig. 1. Multi-scale observations in a real-world cloud workload trace.

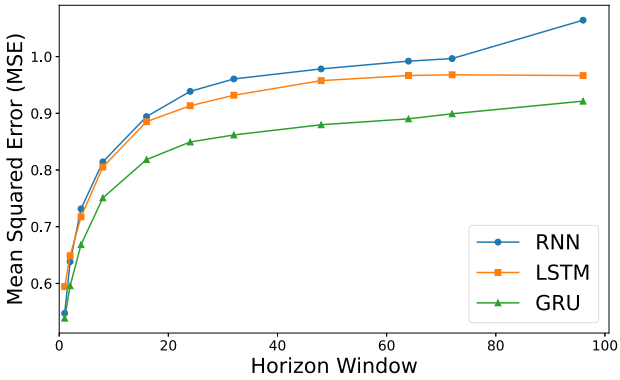


Fig. 2. Prediction accuracy of existing methods across different horizon window sizes.

features [10], demonstrating higher prediction accuracy and learning efficiency compared to regression-based methods.

However, most existing workload prediction methods focus on short-term forecasting of cloud workloads. Compared to short-term forecasting, long-term prediction offers greater strategic value by enabling cloud providers to plan resources and capacity well in advance, avoiding last-minute scaling challenges. It supports proactive infrastructure upgrades, reduces operational costs through better energy management, and aligns long-term business strategies with future demand trends. Although recent works [5], [11] have improved long-term prediction by encoding workload series, their generalization remains limited in handling the complexity and variability of cloud workloads. Meanwhile, general forecasting models like FEDformer [12] and iTransformer [13], designed to capture long-range dependencies using advanced Transformers, struggle with cloud workloads' high dimensionality, heterogeneity, and volatility, leading to performance degradation.

In Fig. 1, we present an example of a cloud workload trace from the Alibaba Cloud Data Center. At a smaller scale, we can observe fluctuations in CPU usage, characterized by a decrease followed by an increase. However, when observing the same cloud workload trace at a larger scale, different characteristics emerge, showing a general upward trend. Cloud workload that exhibits short-term fluctuations shows entirely different patterns over the long term. This uncertainty and high volatility make it challenging for existing models to capture the long-term dynamics of cloud workloads, leading to significant performance degradation in long-term cloud workload predictions as shown in Fig. 2. As the horizon window size gradually increases, the prediction accuracy of RNN-based models significantly declines.

To address these challenges and overcome the limitations of existing methods, we propose a Multi-Scale Network with Convolutions (MSCNet) to capture different patterns and long-term dependencies of cloud workload. Based on our previous observations, we delicately design a multi-scale learning module called Multi-Scale Block (MSB). It consists of three basic modules: Multi-Scale Patch Block (MSPB), Transformer Encoder, and Multi-Scale Convolutions Block (MSCB).

Specifically, we utilize the Multi-Scale Patch Block to segment the original cloud workload sequence into sub-series patches at different scales, where each patch at a particular scale contains information about certain patterns in cloud workloads. This allows us to decompose the original long-term cloud workload sequence into short-term patches at different scales, enabling the model to capture different patterns in cloud workload more easily. Moreover, we extract and learn dependencies between sequences within the same scale of patches in the MSPB. Then, the patches at the same scale are flattened to obtain multiple scale representations of the cloud workload sequence after modeling the patterns and correlation. The Transformer Encoder is employed to effectively extract and learn the relationships between cloud workloads at different scales. Finally, in the Multi-Scale Convolutions Block, we design convolutional operations with varying sizes of kernel, including Equidistant Convolution, Sparse Convolution, and Dense Convolution, to learn and extract intra and inter-scale variation of the cloud workload sequence, better capturing the long-term dependencies and relationships between cloud workload at different scales.

The key contributions of this paper are summarized as follows:

- We propose a Multi-Scale Network with Convolutions (MSCNet) for accurate long-term cloud workload prediction. By extracting and learning features at multiple scales of the original cloud workload, it can effectively capture the dynamic patterns and long-term dependencies of the cloud workload.
- We design the fundamental building block, the Multi-Scale Block, to construct MSCNet, which integrates the Multi-Scale Patch Block, Transformer Encoder, and Multi-Scale Convolutions Block. It adeptly learns long-term relationships within cloud workloads, encompassing both intra and inter-scale variations.
- Extensive experiments on various real-world cloud workload datasets from Alibaba, Google, and Azure cloud data centers demonstrate that our model consistently outperforms existing state-of-the-art approaches.

The remainder of this paper is organized as follows: Section II reviews the related work. In Section III, we provide a detailed description of MSCNet. Section IV evaluates the performance of MSCNet, comparing it with existing prediction methods through experiments on real-world workload datasets. Finally, we conclude the paper and discuss future directions in Section V.

II. Related Work

In this section, we review and summarize existing cloud workload prediction methods and recent research in the field of general time series prediction. The comparison between our proposed approach and related work is presented in Table I.

TABLE I
COMPARSION OF EXSITING WORKLOAD PEDITCITON WORKS AND GENERAL TIME SERIES PREDICTION WORKS

Approach	Prediction Scenario		Technique									Performance Metrics					
	Workload	General	Regression	Machine Learning	Ensemble Learning	Deep Learning						MSE	MAE	RMSE	MAPE	R^2	CDF
						ANN	GAN	MLP	CNN	RNN	Transformer						
Li et al. [6]	✓		✓									✓					
Calheiros et al. [7]	✓		✓												✓		
Yang et al. [8]	✓		✓														
Bi et al. [14]	✓		✓									✓			✓	✓	
Barati et al. [15]	✓			✓								✓	✓	✓	✓		
Nikravesh et al. [16]	✓			✓										✓	✓		
Zhong et al. [17]	✓			✓									✓	✓			
Singh et al. [18]	✓				✓										✓		
CloudInsight [19]	✓				✓									✓	✓		✓
Li et al. [20]	✓					✓								✓			
L-PAW [5]	✓									✓		✓					✓
Start [21]	✓									✓					✓		
EQNN [22]	✓					✓								✓			
esDNN [11]	✓									✓		✓		✓	✓		✓
SG-CBA [23]	✓									✓		✓	✓			✓	
Devi et al. [1]	✓					✓						✓		✓			
VTGAN [24]	✓						✓					✓		✓	✓	✓	
LSTNet [25]		✓								✓							
SegRNN [26]		✓								✓		✓	✓				
Witran [27]		✓								✓		✓	✓				
MICN [28]		✓							✓			✓	✓				
TimesNet [29]		✓							✓			✓	✓				
SCINet [30]		✓							✓			✓	✓				
DLinear [31]		✓						✓				✓	✓				
TSMixer [32]		✓						✓				✓	✓				
TiDE [33]		✓						✓				✓	✓				
Informer [34]		✓									✓	✓	✓				
Autoformer [35]		✓									✓	✓	✓				
FEDformer [12]		✓									✓	✓	✓				
MSCNet (Our work)	✓										✓	✓	✓	✓			

A. Workload Prediction

The existing workload prediction research methods can be roughly divided into two categories: traditional regression-based methods and the latest machine learning-based methods. Traditional methods like the Auto Regressive Integrated Moving Average (ARIMA) model are commonly used to make workload predictions in earlier stages [36]. Li et al. [6] proposed ARIMA-DEC, combining an ARIMA predictor with dynamic error compensation and a time-based billing-aware multi-VM provisioning (TBAMP) algorithm, reducing SLA violations and rental costs. Calheiros et al. [7] presented a proactive ARIMA-based workload prediction for dynamic resource provisioning, achieving high accuracy but facing challenges with peak loads. Yang et al. [8] developed a linear regression-based predictor with an autoscaling mechanism, ensuring SLA compliance while minimizing scaling costs. Bi et al. [14] proposed a hybrid model using Savitzky-Golay filtering, wavelet decomposition, and ARIMA, enhancing prediction accuracy through workload smoothing and trend analysis.

In general, most traditional regression-based workload prediction methods require workloads to have clear patterns or trends to achieve accurate predictions. However, when faced with the non-stationary and high-variance workloads typical of cloud data centers, traditional regression methods often struggle to deliver reasonably accurate predictions. Therefore, recent research has focused primarily on machine learning-based approaches. Barati et al. [15] developed a Tuned Support Vector Regression (TSVR) model using a hybrid Genetic Algorithm and Particle Swarm Optimization (GA-PSO) for optimal parameter selection, enhancing accuracy with a chaotic sequence and validated via Google Cloud simulations. Nikravesh et al. [16] aimed to improve workload prediction accuracy with Support Vector Machine (SVM) and Neural Network (NN) techniques, finding SVM particularly effective for periodic workloads. Zhong et al. [17] introduced a weighted wavelet SVM that integrates wavelet functions and sample weighting, optimized with Particle Swarm Optimization (PSO), achieving better accuracy with Google Cloud data.

In scenarios involving highly dynamic cloud workloads, ensemble models are effective due to their ability to combine multiple predictors and adapt weights dynamically. Singh et al. [18] proposed two ensemble algorithms for large-scale server systems, achieving 89% accuracy for 91% of servers on real and synthetic datasets, effectively handling non-stationary workloads. Kim et al. [19] built on this with CloudInsight, an ensemble framework that adjusts weights based on real-time accuracy. Using multi-class regression with an SVM classifier, CloudInsight outperformed other models.

With the rise of deep learning, numerous efforts have begun to use deep learning-based methods to achieve accurate cloud workload prediction. Li et al. [20] developed a power prediction framework using recursive autoencoders (AE) with fine-grained short-term and coarse-grained long-term forecasting. Chen et al. [5] proposed L-PAW, combining a top-sparse

autoencoder (TSA) and GRU to capture long-term dependencies and predict workloads precisely. Tuli et al. [21] introduced START, an Encoder-LSTM model that mitigates straggler tasks by analyzing resource usage, improving scheduling and reducing SLA violations. Singh et al. [22] advanced this field with EQNN, a quantum neural network using SB-ADE optimization for superior predictions across eight datasets. Xu et al. [11] addressed gradient issues by developing esDNN, enhancing GRU-based predictions and training stability. For edge data centers, Chen et al. [23] proposed SG-CBA, combining Savitzky-Golay filtering, CNN, BiLSTM, and attention mechanisms to ensure accurate predictions with real-world data. Devi et al. [1] introduced a hybrid ARIMA-ANN model to capture both linear and nonlinear trends, refined with Savitzky-Golay filtering for better resource management on Google and BitBrain datasets. Maiyza et al. [24] developed VTGAN, a hybrid model that dynamically adjusts prediction steps and sliding windows, outperforming traditional methods in workload and trend forecasting.

Although some works, such as CloudInsight [19], L-PAW [5], EQNN [22], and esDNN [11], attempt to predict long-term dependencies beyond short-term periodicity through implicit modeling, they are limited by single-scale modeling and learning, failing to effectively identify the various patterns in cloud workloads.

B. General Time Series Prediction

Time series prediction has gained significant attention across various fields due to its importance, leading to the development of numerous general-purpose forecasting models. These existing works can be broadly categorized into RNN-based, Convolutional Neural Network (CNN)-based, Multilayer Perceptron (MLP)-based, and Transformer-based approaches.

RNN-based models are widely used in time series forecasting tasks due to their efficiency in learning sequential patterns. However, the sequential nature of RNNs makes it challenging to capture long-term dependencies, which limits their performance in long-range forecasting tasks. Notable studies in this area include LSTNet [25], SegRNN [26], and WITRAN [27].

CNN-based approaches excel in time series forecasting by effectively capturing local patterns and extracting relevant features. Recent models such as MICN [28], TimesNet [29], and SCINet [30] demonstrate impressive performance in long-range forecasting by stacking multiple convolutional layers to capture dependencies over longer horizons [37].

MLP-based methods have recently gained popularity in time series forecasting due to their lightweight architectures and efficient design strategies [38]. Models such as DLinear [31], TSMixer [32], and TiDE [33] have achieved promising results. However, their performance may be limited in complex, dynamic scenarios such as cloud workload forecasting.

Inspired by the success of Transformers in natural language processing tasks, many recent studies have adopted Transformer-based methods for time series forecasting. The self-attention mechanism in Transformers enables effective modeling of long-term dependencies within time series data. Unlike RNN-based approaches, which rely on sequential processing, the self-attention mechanism processes sequences in parallel, allowing each element to attend to any other element in the sequence. Recent works, such as Informer [34], Autoformer [35], and FEDformer [12], have achieved state-of-the-art performance in long-range forecasting tasks.

Overall, most existing workload prediction methods focus on short-term prediction of cloud workloads and face challenges in achieving accurate long-term predictions. Although some studies in general time series prediction have focused on long-range predictions [12], [29] [28], there is little discussion or experimentation in the context of long-term prediction for cloud workload. Compared to commonly explored forecasting scenarios such as weather, traffic, and electricity, cloud workloads exhibit higher dimensionality and greater volatility, posing significant challenges for general-purpose time series models in achieving accurate long-term predictions in cloud environments. To overcome the limitations of existing methods, by extracting and learning features at different scales from the long-term cloud workload series, MSCNet effectively identifies and models different patterns and long-term dependencies in cloud workload, thereby achieving more accurate long-term predictions.

III. Model

A. Problem Formulation

Cloud workload prediction involves forecasting future cloud workloads based on historical workloads. Specifically, the task is to predict T future values $Y = (y_1, \ldots, y_T)$ from L historical values $X = (x_1, \ldots, x_L)$, where the cloud workload has D dimensions, each representing a resource utilization metric (e.g., CPU utilization, memory utilization, etc.). To better describe the task and model details, we refer to L as the look-back window size and T as the prediction horizon. Long-term cloud workload prediction differs from short-term prediction by requiring a larger look-back window and prediction horizon, making it more challenging for the model to achieve accurate predictions.

B. Model Architecture

MSCNet is a powerful model capable of long-term cloud workload prediction by leveraging multi-scale modeling of the original sequences and using a vanilla Transformer encoder as its core to enhance prediction accuracy. The overall architecture of MSCNet is shown in Fig. 3. Overall, MSCNet can be divided into two parts: the Trend Prediction Block and the Multi-Scale Prediction Block.

The Trend Prediction Block is primarily used to extract and learn trend information from highly fluctuating cloud workloads, enabling it to predict future changes in the workload. Meanwhile, the Multi-Scale Prediction Block performs multi-scale modeling of the original cloud workload to capture various patterns and long-term dependencies, thereby extracting and learning the overall information of the cloud workload sequences. By integrating these two modules, we can achieve a more accurate long-term prediction of cloud workloads.

Additionally, to reduce the impact of volatility and noise in cloud workload data on the model's performance and to better capture the relationships between different cloud workloads, we perform some data preprocessing tasks. These tasks mainly

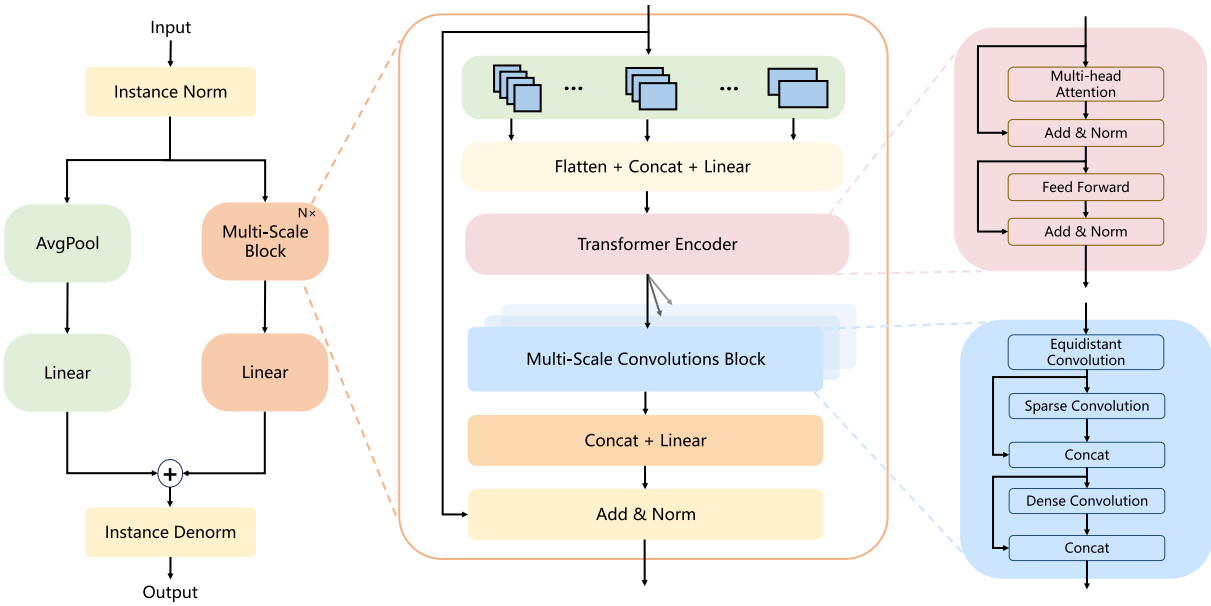


Fig. 3. Overall architecture of MSCNet.

include two parts: reversible instance normalization (RevIN) and the channel independent strategy. These two modules enable the model to better learn the different patterns and long-term dependencies in the cloud workload data, thereby achieving more accurate long-term cloud workload predictions. We provide a more detailed explanation in the Section III-C.

The overall process of cloud workload prediction using MSCNet can be represented as (1). More detailed explanations will be provided in Sections III-D and III-E respectively.

$$\begin{aligned} X_r &= \text{RevIN}(X), \\ Y_t &= \text{Trend Prediction Block}(X_r), \\ Y_m &= \text{Multi-Scale Prediction Block}(X_r), \\ Y &= \text{RevIN}^{-1}(Y_t + Y_m). \end{aligned} \tag{1}$$

C. Data Preprocessing

Instance Normalization: To enhance the stability of the cloud workload sequences, inspired by the work of Kim et al. [39], we use reversible instance normalization (RevIN) to mitigate the impact of data distribution drift on model performance. The entire process of RevIN can be formulated as follows:

$$\begin{aligned} \text{E}[X^{(i)}] &= \frac{1}{L}\sum_{t=1}^{L} X_t^{(i)}, \\ \text{Var}[X^{(i)}] &= \frac{1}{L}\sum_{t=1}^{L}(X_t^{(i)} - \text{E}[X^{(i)}])^2, \\ X_r^{(i)} &= \frac{X^{(i)} - \text{E}[X^{(i)}]}{\sqrt{\text{Var}[X^{(i)}] + \varepsilon}}, \\ Y^{(i)} &= \sqrt{\text{Var}[X^{(i)}] + \varepsilon} \cdot Y_r^{(i)} + \text{E}[X^{(i)}], \end{aligned} \tag{2}$$

where $X^{(i)} \in \mathbb{R}^{1\times L}$ represents the ith dimension of the historical cloud workload, and $Y^{(i)} \in \mathbb{R}^{1\times T}$ represents the ith dimension of the future cloud workload. The instance normalization of X yields X_r, and the inverse instance normalization of Y_r results in Y.

Channel Independent Strategy: Existing cloud workload methods typically use a channel dependent strategy in modeling, mapping the channel dimension to hidden layer representations. This approach focuses more on learning the relationships between different channels of the cloud workload. However, recent studies [40], [41] have shown that adopting a channel independent strategy in modeling can effectively improve model prediction performance and robustness, reducing the impact of data distribution drift. Simply put, a channel independent strategy involves modeling each channel separately, using the historical values of each channel to predict future values. For models using the channel dependent strategy, the input data has the dimension $X_{CD} \in \mathbb{R}^{L\times D}$, whereas for models using the channel independent strategy, the input data has the dimension $X_{CI} \in \mathbb{R}^{L\times 1}$ as shown in Fig. 4. Therefore, unless otherwise specified, the variables mentioned hereafter in Section III-E all refer to a single channel of cloud workload.

D. Trend Prediction Block

The original cloud workload is characterized by strong volatility and high variance, containing noise and outliers that affect prediction accuracy. Inspired by existing prediction studies [14], [42] [35], we further extract the trend component of the original cloud workload. The Trend Prediction Block is used to learn the trend vector of the original cloud workload. By using moving averages, we smooth out periodic fluctuations and highlight the long-term trend of the sequence. Finally, a prediction of future trends is achieved through a simple linear layer. The necessity of modeling the trend component for non-stationary

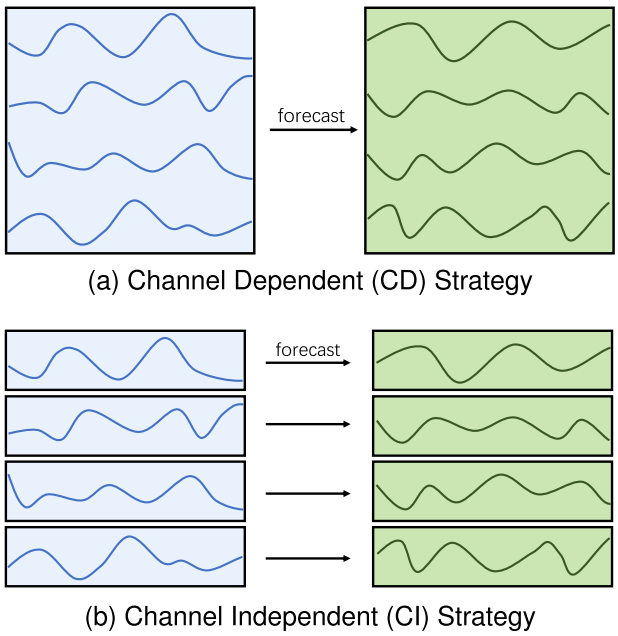


Fig. 4. Comparison of channel dependent and channel independent strategies.

cloud workload prediction is demonstrated in Section IV-C. This process can be computed using the following equation:

$$X_t = \text{AvgPool}(\text{Padding}(X_r))_{\text{kernel}},$$
$$Y_t = \text{Linear}(X_t), \tag{3}$$

where we use the average pooling with padding to obtain the trend component with unchanged length. Y_t denotes the output of the Trend Prediction Block, and the kernel size matches one of the scales in the Multi-Scale Prediction Block.

E. Multi-Scale Prediction Block

As shown in Fig. 3, the Multi-Scale Prediction Block is designed to model comprehensive cloud workload sequences. To effectively capture various patterns and long-term dependencies, we develop its core component—the Multi-Scale Block. Each Multi-Scale Prediction Block contains N Multi-Scale Blocks. The input to the nth Multi-Scale Block is defined as $X_m^{(n)} \in \mathbb{R}^{D\times L}$, and the output as $Y_m^{(n)} \in \mathbb{R}^{D\times L}$, where $n \in (1, N)$.

Each Multi-Scale Block comprises three sub-modules: the Multi-Scale Patch Block, the Transformer Encoder, and the Multi-Scale Convolutions Block. Through the seamless integration of these modules, the Multi-Scale Prediction Block enables precise long-term forecasting of cloud workloads.

Multi-Scale Patch Block: To effectively learn different patterns in cloud workloads, we design the Multi-Scale Patch Block. Its core idea is to convert a univariate cloud workload series into S cloud workload series at different scales.

Specifically, as shown in Fig. 5, for a given scale s, the original univariate cloud workload series $X_m \in \mathbb{R}^{1\times L}$ is duplicated S times to obtain multivariate cloud workload series $X_p \in \mathbb{R}^{S\times L}$. Each series $X_p^{(s)} \in \mathbb{R}^{1\times L}$ within multivariate cloud workload series is then divided into K_s subsequences $X_p^{(s,k)} \in \mathbb{R}^{1\times P_s}$ according to the patch length P_s, where $s \in (1, S)$, $K_s = \frac{L}{P_s}$ and $k \in (1, K_s)$. Ultimately, the original univariate cloud workload series is converted into a multi-scale patch time series. This process can be represented by the following equation:

$$X_p = \text{Duplicate}(X_m)_{\text{times}=S},$$
$$X_p^{(s,1)}, \ldots, X_p^{(s,K_s)} = \text{Reshape}(X_p^{(s)})_{\text{length}=P_s}. \tag{4}$$

Furthermore, to capture the relationships between patch subsequences at various scales in the multi-scale patch time series, we use linear layer mapping to learn the intra-patch variation. The mapped sequences are then flattened and concated to transform them into multi-scale time series X_e. It can be formulated as the following equation:

$$\hat{X}_p^{(s,k)} = \text{Linear}(X_p^{(s,k)}),$$
$$\hat{X}_m^{(s)} = \text{Flatten}(\hat{X}_p^{(s,1)}, \ldots, \hat{X}_p^{(s,K_s)}),$$
$$X_e = \text{Concat}(\hat{X}_m^{(1)}, \ldots, \hat{X}_m^{(S)}). \tag{5}$$

Transformer Encoder: Compared to RNNs, which process information sequentially and have a long information path prone to vanishing gradients, Transformers use attention to access all positions in parallel, shortening the information path and better capturing long-range dependencies. Therefore, to effectively learn and extract features of cloud workloads and the relationships between multi-scale series, we use a vanilla Transformer encoder to map the multi-scale time series $X_e \in \mathbb{R}^{S\times L}$ to hidden layer representations. Specifically, we employ a linear projection to transform $X_e \in \mathbb{R}^{S\times L}$ into $X_d \in \mathbb{R}^{S\times d}$, which serves as the input to the Transformer encoder, and d denotes the dimension of the hidden layer. Each head $h = 1, \ldots, H$ in the multi-head attention mechanism then converts this input into query matrices $\mathbf{Q}_h = X_d\mathbf{W}_h^Q$, key matrices $\mathbf{K}_h = X_d\mathbf{W}_h^K$, and value matrices $\mathbf{V}_h = X_d\mathbf{W}_h^V$, where $\mathbf{W}_h^Q$, $\mathbf{W}_h^K \in \mathbb{R}^{d\times d_k}$ and $\mathbf{W}_h^V \in \mathbb{R}^{d\times d}$. After this, a scaled dot-product operation is used to obtain the attention output $\mathbf{O}_h \in \mathbb{R}^{S\times d}$:

$$\mathbf{O}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h),$$
$$= \text{Softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{d_k}}\right)\mathbf{V}_h. \tag{6}$$

In the Transformer encoder, batch norm layers and a residual connection with a feedforward network are also included, as shown in Fig. 3. After this, we obtain its representation denoted as $\hat{X}_d \in \mathbb{R}^{S\times d}$.

Multi-Scale Convolutions Block: To better extract different patterns and cycles of cloud workloads, we adopt C multi-scale convolution blocks with kernel sizes kernel_c where $c \in (1, C)$. These blocks mainly consist of Equidistant Convolution, Sparse Convolution, and Dense Convolution, as shown in Fig. 3.

We use Equidistant Convolution and Sparse Convolution to extract local information about cloud workloads, and Dense Convolution to extract global information about cloud workloads. More specifically, for Equidistant Convolution, we set its kernel size to kernel_c, padding to 0, and stride to 1. For Sparse Convolution, we set its kernel size to kernel_c, and both padding

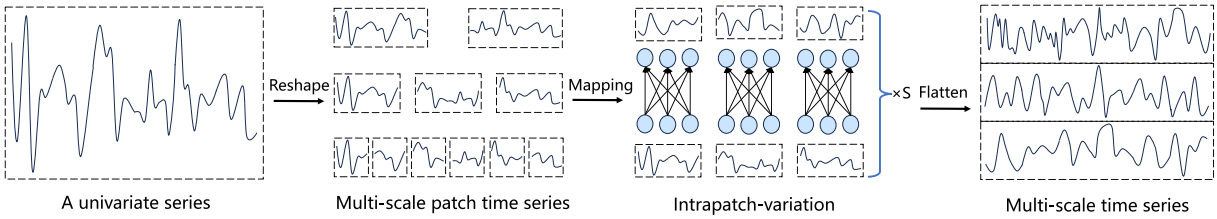


Fig. 5. An example of a univariate series to illustrate how the Multi-Scale Patch Block operates.

and stride to $\frac{\text{kernel}_c}{2}$. For Dense Convolution, we set its kernel size and stride to $\frac{\text{kernel}_c}{2}$, and padding to 0. The entire process for the cth Multi-Scale Convolutions Block can be formulated as the following equations:

$$\begin{aligned} X_{equi}^{(c)} &= \text{Equidistant Conv}(\hat{X}_d)_{\text{kernel}_c}, \\ X_{sparse}^{(c)} &= \text{Sparse Conv}(\text{Concat}(\hat{X}_d, X_{equi}^{(c)}))_{\text{kernel}_c}, \\ X_{dense}^{(c)} &= \text{Dense Conv}(\text{Concat}(X_{equi}^{(c)}, X_{sparse}^{(c)}))_{\frac{\text{kernel}_c}{2}}, \\ X_c^{(c)} &= \text{Concat}(X_{sparse}^{(c)}, X_{dense}^{(c)}), \end{aligned} \tag{7}$$

where $X_c^{(c)}$ represents the output of the cth Multi-Scale Convolutions Block.

To aggregate the outputs of different scales from the Multi-Scale Convolutions Block, we use concatenation and a linear layer to combine and map the outputs of each Multi-Scale Convolutions Block. And then a batch norm layer is used to obtain the output of the nth Multi-Scale Block as well as the input of the $(n+1)$th Multi-Scale Block, $Y_m^{(n)} \in \mathbb{R}^{D \times L}$, as shown in Fig. 3.

After extracting and learning the features of cloud workloads through N Multi-Scale Blocks, we use a simple linear layer to map $Y_m^{(N)}$ to obtain the final output of the Multi-Scale Prediction Block, $Y_m \in \mathbb{R}^{D \times L}$. A more complete and detailed overall process of the Multi-Scale Prediction Block is presented in Algorithm 1.

Overall, the combination of the Trend Prediction Block and the Multi-Scale Prediction Block can effectively mitigate the performance interference caused by the high volatility and noise of cloud workloads. The design of the Multi-Scale Block enables the model to effectively extract and learn the different periodic patterns and long-term dependencies of historical cloud workloads at different scales, achieving accurate prediction of long-term dynamic changes in cloud workloads. The key steps of MSCNet are shown in Algorithm 2.

IV. Experiments

To evaluate our proposed model, we conduct extensive experiments with real-world cloud workload datasets. In this section, we will provide a detailed description of the cloud workload datasets, baseline methods, evaluation metrics, and experimental results, showing how MSCNet outperforms existing prediction methods. All experiments were executed in PyTorch on an NVIDIA 4090 GPU with 24 GB of memory.

A. Experimental Setup

Datasets: We utilize three real-world cloud workload datasets from Alibaba, Google, and Azure to validate our model.

- *Alibaba Cluster Traces [43]:* The Alibaba Cluster Traces dataset, designated as cluster-trace-v2018, encompasses data from approximately 4,000 machines over 8 days. Collected in 2018, this dataset includes both long-running applications and batch cloud workloads, with a sampling interval of 10 seconds. The data can be accessed on GitHub.[1]
- *Google Cluster Traces [44]:* The Google Cluster Traces dataset provides detailed operational data from a cluster of around 12,500 machines, recorded over 29 days in May 2011. This dataset, referred to as cluster-2011-2, offers insights into various cluster performance metrics, with a sampling interval of 5 minutes. The data is available on GitHub.[2]
- *Azure Traces [45]:* The Azure Traces dataset contains a representative subset of the first-party Azure VM workload in one geographical region, denoted as Azure Public Dataset V1. It collects data from approximately 2,000,000 VMs over thirty days (such as CPU utilization, VM information, etc.), with a sampling interval of 5 minutes. The data can be accessed on GitHub.[3]

Given the importance of CPU utilization in cloud data centers and the cloud workload metrics collected by various datasets, we use CPU utilization as the input to our model to evaluate its prediction accuracy and performance. Additionally, we randomly select 1,000 machines from each dataset and plot the data distribution of CPU utilization for each dataset to facilitate statistical analysis, as shown in Fig. 6. As observed, there are substantial differences in data distributions across the three datasets: the Azure dataset has a high concentration of CPU utilization below 0.1, while the Google dataset is primarily distributed around 0.2. In contrast, the Alibaba dataset exhibits a more uniform distribution, with values concentrated around 0.3. Then, we randomly select 100 machines to extract their CPU utilization for our model evaluation dataset. All three datasets are divided into training, testing, and validation sets in a 7:1:2 ratio.

Baselines: To validate the effectiveness of our approach, we compare MSCNet with several existing prediction methods that exhibit state-of-the-art performance.

[1]https://github.com/alibaba/clusterdata
[2]https://github.com/google/cluster-data
[3]https://github.com/Azure/AzurePublicDataset

Algorithm 1: Multi-Scale Prediction Block.

Input: preprocessed cloud workload $X_r \in \mathbb{R}^{D\times L}$
Output: cloud workload representation $Y_m \in \mathbb{R}^{D\times L}$
Initialize: number of Multi-Scale Blocks N, number of scales in Multi-Scale Patch Block S, patch length P_s, number of patches $K_s = \frac{L}{P_s}$, number of Multi-Scale Convolutions Blocks C
for *each Multi-Scale Block* $n \in \{1, 2, \ldots, N\}$ **do**
  for *each dimension of cloud workload* $i \in \{1, 2, \ldots, D\}$ **do**
    Calculate the n-th Multi-Scale Block's input;
    if $n = 1$ **then**
      $X_m^{(n)} \Leftarrow X_r$;
    else
      $X_m^{(n)} \Leftarrow Y_m^{(n-1)}$;
    end
    for *each scale in Multi-Scale Patch Block* $s \in \{1, 2, \ldots, S\}$ **do**
      Calculate the multi-scale patch time series at scale s with Eq. (4);
      for *each patch in the multi-scale patch time series* $k \in \{1, 2, \ldots, K_s\}$ **do**
        Learn the intra-patch variation in the patch with Eq. (5);
      end
    end
    Integrate patches into multi-scale time series with Eq. (5);
    Calculate the Transformer encoder output representation $\hat{X}_d$ with Eq. (6);
    for *each Multi-Scale Convolutions Block* $c \in \{1, 2, \ldots, C\}$ **do**
      Extract and learn features from cloud workload representation with Eq. (7);
    end
    Calculate the n-th Multi-Scale Block's output;
    $X_c \Leftarrow \text{Linear}(\text{Concat}(X_c^{(1)}, \ldots, X_c^{(C)}))$;
    $Y_m^{(n)} \Leftarrow \text{Add \& Norm}(X_m^{(n)}, X_c)$;
  end
end
Calculate the Multi-Scale Prediction Block's output;
$Y_m \Leftarrow \text{Linear}(Y_m^{(N)})$;
return Y_m;

Algorithm 2: Multi-Scale Network with Convolutions (MSCNet) for Workload Prediction.

Input: historical cloud workload $X \in \mathbb{R}^{L\times D}$
Output: future cloud workload $Y \in \mathbb{R}^{T\times D}$
Initialize: learning rate, learning rate decay, batch size, dropout rate, early stopping patience, number of training epochs N_e, look-back window size L, horizon window size H, workload dimension D, hidden units d
for *each training epoch* $n \in \{1, 2, \ldots, N_e\}$ **do**
  Utilize reversible instance normalization to preprocess cloud workload;
  for *each dimension of cloud workload* $i \in \{1, 2, \ldots, D\}$ **do**
    $X_r^{(i)} \Leftarrow \frac{X^{(i)} - \mathbb{E}[X^{(i)}]}{\sqrt{\text{Var}[X^{(i)}] + \varepsilon}}$;
  end
  Calculate the Trend Prediction Block's output;
  $X_t \Leftarrow \text{AvgPool}(\text{Padding}(X_{ci}))_{\text{kernel}}$;
  $Y_t \Leftarrow \text{Linear}(X_t)$;
  Call Algorithm 1 to obtain the Multi-Scale Prediction Block's output;
  $Y_m \Leftarrow \text{Multi-Scale Prediction Block}(X_{ci})$;
  Aggregate the outputs of the two above blocks;
  $Y_r \Leftarrow Y_t + Y_m$;
  for *each dimension of cloud workload* $i \in \{1, 2, \ldots, D\}$ **do**
    $Y^{(i)} \Leftarrow \sqrt{\text{Var}[X^{(i)}] + \varepsilon} \cdot Y_r^{(i)} + \mathbb{E}[X^{(i)}]$;
  end
end
return Y;

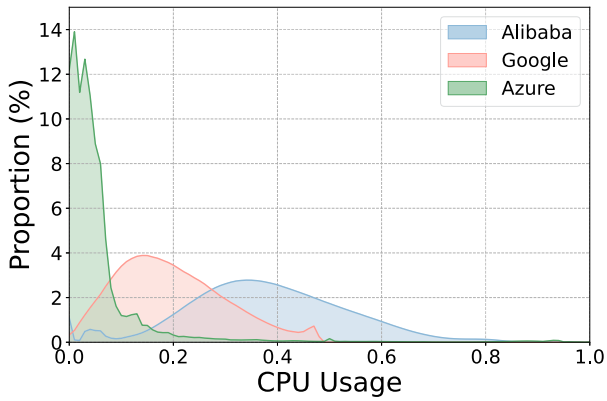


Fig. 6. Data distribution across different datasets.

- *L-PAW [5]:* This cloud workload prediction method leverages top-sparse auto-encoders (TSA) and gated recurrent units (GRU) for accurate prediction of high-dimensional cloud workloads. TSA is designed to learn and extract representations of high-dimensional and high-variance cloud workloads, while the GRU module captures long-term dependencies.
- *esDNN [11]:* A GRU-based cloud workload prediction method that uses a sliding window to convert multivariate data into a supervised learning time series. It employs a modified GRU to achieve precise workload prediction.
- *SG-CBA [23]:* A BiLSTM-based cloud workload prediction method that integrates the Savitzky-Golay filter, CNN, attention mechanism, and BiLSTM for accurate workload prediction. The Savitzky-Golay filter smooths workload data and the deep learning module combines CNN and

TABLE II
PARAMETERS USED IN MODEL TRAINING AND TUNING

Parameter	Value
Look-back window size (L)	96
Horizon window size (H)	{24, 48, 72, 96}
Loss function	L2
Optimizer	ADAM
Initial learning rate	1×10^{-4}
Learning rate decay	0.5
Batch size	32
Dropout rate	0.2
Number of hidden units	128
Training epochs	100
Early stopping patience	3
Number of Multi-Scale Blocks (N)	3
Patch size (S)	5
Patch length (P)	{16, 24, 32, 48, 96}
Number of Multi-Scale Convolution Blocks (C)	3

BiLSTM with an attention mechanism for precise workload forecasting.

- *FEDformer [12]:* This Transformer-based prediction method combines the Transformer model with seasonal-trend decomposition. The decomposition method captures the global outline of the time series, while the Transformer captures more detailed structures. By using a frequency-enhanced Transformer, it achieves more efficient and accurate time series forecasting.
- *MICN [28]:* A convolution-based prediction method that combines local and global features to capture the overall view of the time series. It uses down-sampling convolution and equidistant convolution to effectively extract local features and global correlations of the sequence, resulting in more accurate and efficient time series predictions.
- *TimesNet [29]:* This method transforms one-dimensional time series into a set of multi-period two-dimensional tensors for learning and modeling. It uses TimesBlock as its core to efficiently and adaptively discover periodicity and extract complex temporal variations, enabling accurate forecasting.

Metrics: To evaluate the predictive accuracy and performance of the model, we use three commonly used evaluation metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). The specific metrics are defined by the following formulas:

$$\text{MSE} = \frac{1}{n \times H} \sum_{i=1}^{n} \sum_{t=1}^{H} (y_{i,t} - \hat{y}_{i,t})^2,$$

$$\text{MAE} = \frac{1}{n \times H} \sum_{i=1}^{n} \sum_{t=1}^{H} |y_{i,t} - \hat{y}_{i,t}|,$$

$$\text{RMSE} = \sqrt{\frac{1}{n \times H} \sum\nolimits_{i=1}^{n} \sum\nolimits_{t=1}^{H} (y_{i,t} - \hat{y}_{i,t})^2}. \quad (8)$$

The MSE measures the average squared difference between actual and predicted values, making it sensitive to large errors. The MAE measures the average absolute difference, providing a linear score that is more robust to outliers. The RMSE is the square root of the average squared differences, combining properties of both MSE and standard deviation to indicate the spread of residuals. Lower values for these metrics signify better performance.

Configuration: To evaluate the performance of various models in long-term cloud workload prediction, we use a look-back window size of $L = 96$ and a horizon window size H set to {24, 48, 72, 96}. All models are trained using the L2 loss function and the ADAM optimizer. We initialize the learning rate at 1×10^{-4} and apply a learning rate decay factor of 0.5. The batch size is set to 32, the dropout rate to 0.2, the number of hidden units to 128, and the training proceeds for up to 100 epochs with early stopping, using a patience of 3 epochs. For the model's hyperparameters, we employ a heuristic search algorithm to determine optimal values. The selected configuration is shown in Table II.

B. Experimental Results

To compare the long-term performance of MSCNet with other methods in cloud workload forecasting, we analyze their prediction accuracy using three performance metrics, as shown in Table III, which includes data from Alibaba, Google, and Azure. We observe that existing RNN-based workload prediction methods perform poorly in long-term cloud workload prediction due to issues such as gradient explosion/vanishing and error accumulation. Among them, SG-CBA performs best thanks to its attention mechanism and the integration of CNN with the Savitzky-Golay filter. TimesNet, MICN, and FEDformer outperform RNN-based methods in accuracy, demonstrating stronger capabilities in capturing and learning long-term dependencies of workloads. MICN excels on the Alibaba and Google datasets, while TimesNet performs better on the Azure dataset. It is worth noting that our proposed MSCNet surpasses both MICN and TimesNet, significantly outperforming existing methods. Compared to existing workload prediction methods, MSCNet improves MAE by an average of 14.1%, 55.7%, and 43.1% on the Alibaba, Google, and Azure datasets, respectively. It achieves the highest accuracy across three metrics for different prediction windows on all datasets, demonstrating its robust ability to model long-term dependencies and periodic patterns at multiple scales.

Next, to further compare the performance of different models at various prediction lengths, we plot the MSE, MAE, and RMSE performance metrics on the Google dataset, as shown in Fig. 7. Fig. 7 displays the curves of different models' MSE, MAE, and RMSE on the Google dataset as the prediction length varies. It can be seen that compared to other methods, MSCNet achieves better performance across all three metrics. Additionally, as the prediction length increases, the performance gap between our model and the other models gradually widens, demonstrating better long-term cloud workload prediction capabilities.

Fig. 8 illustrates the distribution of MAE for different models across three cloud workload datasets using violin plots. The shapes of the violin plots vary slightly across the different datasets due to the unique distributions and characteristics of each dataset. However, within the same dataset, the shapes of

TABLE III
COMPARISON OF DIFFERENT MODELS ON THREE DATASETS ACROSS VARIOUS HORIZONS USING MSE, MAE, AND RMSE METRICS

Models		MSCNet			TimesNet			MICN			FEDformer			SG-CBA			esDNN			L-PAW		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	**0.5687**	**0.4565**	**0.7541**	0.6457	0.5029	0.8036	0.5993	0.4704	0.7741	0.6671	0.5265	0.8168	0.9347	0.6369	0.9668	0.9672	0.6508	0.9835	0.9669	0.6489	0.9833
	48	**0.6274**	**0.4986**	**0.7921**	0.6962	0.5332	0.8344	0.6452	0.5041	0.8032	0.6992	0.5453	0.8362	0.9570	0.6457	0.9783	1.0198	0.6753	1.0098	0.9853	0.6570	0.9926
	72	**0.6622**	**0.5215**	**0.8138**	0.7134	0.5467	0.8446	0.6742	0.5235	0.8211	0.7241	0.5592	0.8510	0.9609	0.6466	0.9803	1.0356	0.6822	1.0176	0.9915	0.6599	0.9957
	96	**0.6892**	**0.5379**	**0.8302**	0.7369	0.5539	0.8584	0.6973	0.5382	0.8350	0.7474	0.5722	0.8645	0.9705	0.6510	0.9852	1.0706	0.7019	1.0343	1.0620	0.6956	1.0305
Google	24	**1.0371**	**0.7290**	**1.0184**	1.2894	0.8260	1.1355	1.2381	0.8174	1.1127	1.3507	0.8568	1.1622	2.7016	1.2866	1.6436	2.9454	1.3485	1.7162	2.8200	1.3229	1.6793
	48	**1.2555**	**0.8087**	**1.1205**	1.4768	0.8858	1.2152	1.3744	0.8801	1.1724	1.4706	0.8939	1.2127	2.8830	1.3352	1.6978	3.1979	1.4129	1.7883	3.0607	1.3832	1.7495
	72	**1.4151**	**0.8640**	**1.1896**	1.6046	0.9309	1.2667	1.6842	1.0071	1.2978	1.5778	0.9293	1.2561	3.1151	1.3937	1.7647	3.2486	1.4279	1.8023	3.1647	1.4089	1.7790
	96	**1.5288**	**0.9051**	**1.2365**	1.7195	0.9666	1.3113	1.8513	1.0592	1.3606	1.6595	0.9573	1.2882	3.1726	1.4087	1.7811	3.2963	1.4374	1.8156	3.1814	1.4141	1.7836
Azure	24	**4.0681**	**0.4400**	**2.0170**	4.6673	0.5104	2.1604	4.6865	0.5528	2.1648	4.8419	0.6369	2.2004	9.2718	0.8958	3.0449	9.2404	0.8843	3.0398	9.2538	0.8950	3.0420
	48	**4.3165**	**0.4546**	**2.0776**	4.8019	0.5206	2.1913	4.7784	0.5599	2.1859	4.8982	0.5798	2.2132	9.3934	0.8860	3.0648	9.3310	0.8835	3.0547	9.3454	0.9071	3.0570
	72	**4.4485**	**0.4725**	**2.1091**	4.9017	0.5295	2.2140	5.1306	0.6196	2.2651	4.9955	0.5851	2.2351	9.4553	0.8889	3.0749	9.4170	0.8954	3.0687	9.4086	0.9104	3.0673
	96	**4.5562**	**0.4776**	**2.1345**	4.9610	0.5329	2.2273	5.3403	0.6374	2.3109	5.0972	0.6010	2.2577	9.5279	0.8963	3.0867	9.4965	0.9163	3.0816	9.4901	0.9112	3.0806

The horizon window size $H \in \{24, 48, 72, 96\}$ is set for all datasets.

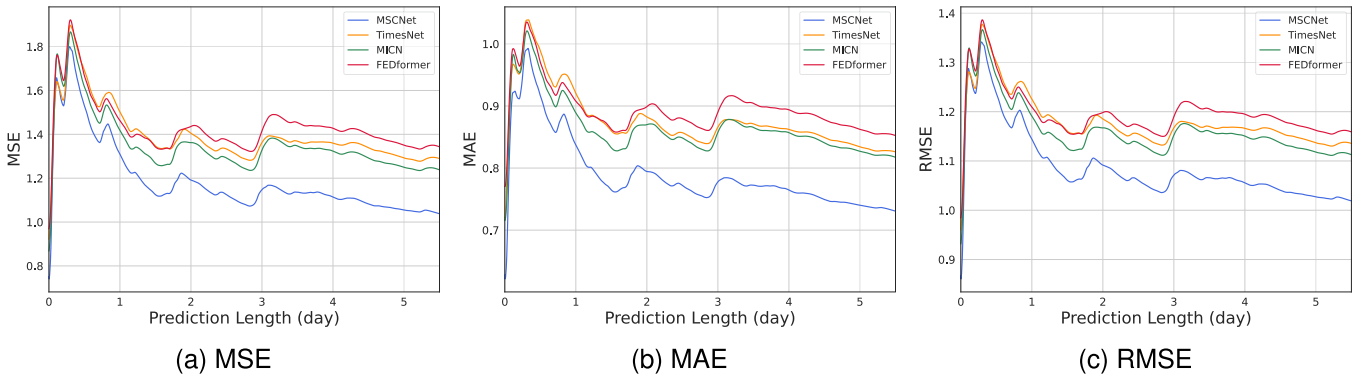


Fig. 7. Deatiled MSE, MAE, and RMSE results of different models with various prediction lengths on the Google dataset.

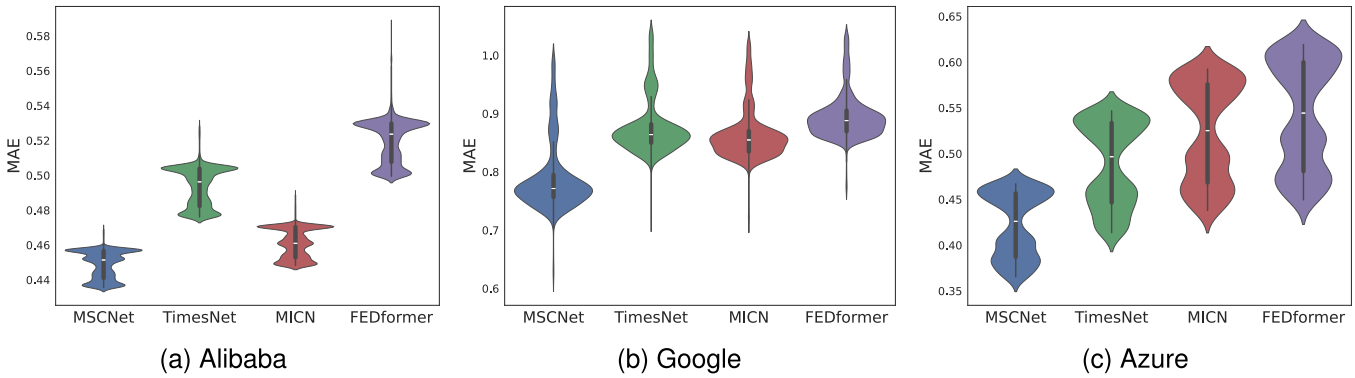
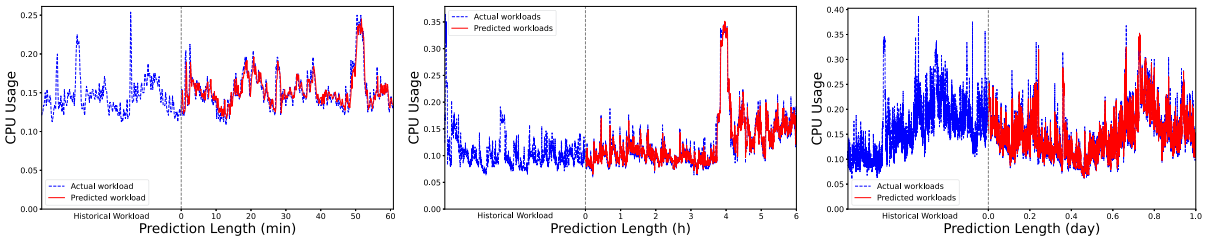


Fig. 8. The distribution of MAE for different models on three cloud workload datasets.

the violin plots for the various models are relatively similar. Compared to other models, MSCNet exhibits a smaller median MAE, indicating higher predictive accuracy. Its narrower interquartile range and more concentrated distribution suggest better robustness. Additionally, MSCNet shows a smaller maximum MAE on the Alibaba dataset and a smaller minimum MAE on the Google dataset, highlighting its superior performance and stability across different datasets. In Fig. 9, we present examples of MSCNet's predictions on the Alibaba dataset across different prediction lengths. For minute-level workload predictions, MSCNet accurately captures trends and dynamic changes, resulting in precise load forecasts. At the hour level, MSCNet effectively adapts to and predicts sudden peaks and fluctuations. For day-level predictions, MSCNet successfully learns long-term dependencies and periodic trends in the workload.

To compare the transferability and generalization capability of MSCNet with other baseline methods, we conduct transfer experiments on the Google and Azure datasets. In the context of evaluating cross-dataset transferability, the models are initially trained on the Alibaba dataset and then directly used to predict the Google and Azure datasets. Table IV presents the evaluation results of our transfer learning experiments. As shown, MSCNet achieves the best or second-best performance in all settings, surpassing other baseline methods and highlighting its robust generalization and transferability. This is attributed to one of MSCNet's key advantages: modeling the raw cloud workload



(a) Minute-level (b) Hour-level (c) Day-level

Fig. 9. Prediction performance display of MSCNet on the Alibaba dataset with various levels of prediction length.

TABLE IV
THE EXPERIMENTAL RESULTS OF TRANSFERRING MODELS TRAINED ON THE ALIBABA DATASET TO THE GOOGLE AND AZURE DATASETS

Models		MSCNet			TimesNet			MICN			FEDformer			SG-CBA			esDNN			L-PAW		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Google	24	**1.0543**	**0.7372**	**1.0268**	1.3449	0.8507	1.1597	1.0601	0.7601	1.0296	1.3646	0.8585	1.1682	2.4619	1.2327	1.5690	2.7421	1.3000	1.6559	2.6416	1.2642	1.6253
	48	1.2760	**0.8189**	1.1296	1.5311	0.9084	1.2374	**1.2401**	0.8263	**1.1136**	1.5091	0.9051	1.2285	2.5414	1.2487	1.5942	2.6857	1.2905	1.6388	2.6685	1.2803	1.6336
	72	1.4348	0.8732	1.1978	1.6567	0.9520	1.2871	**1.3620**	**0.8713**	**1.1671**	1.6217	0.9424	1.2735	2.5545	1.2564	1.5983	3.1804	1.4133	1.7834	2.6754	1.2872	1.6357
	96	1.5480	0.9133	1.2442	1.7998	0.9953	1.3416	**1.4544**	**0.9059**	**1.2060**	1.7063	0.9723	1.3062	2.6026	1.2690	1.6132	2.8458	1.3401	1.6869	2.9148	1.3437	1.7073
Azure	24	**4.5138**	**0.4920**	**2.1246**	4.8229	0.5448	2.1961	4.5344	0.5224	2.1294	4.7193	0.5558	2.1724	9.1484	1.2360	3.0246	9.2822	1.2470	3.0467	9.2074	1.1064	3.0344
	48	4.8271	**0.5170**	2.1971	4.9764	0.5624	2.2308	**4.7913**	0.5444	**2.1889**	4.8695	0.5622	2.2067	9.3278	1.2737	3.0542	9.3195	1.1918	3.0528	9.2778	1.2007	3.0460
	72	**4.9291**	**0.5308**	**2.2202**	5.0224	0.5552	2.2411	4.9418	0.5597	2.2230	4.9779	0.5695	2.2311	9.5357	1.2723	3.0880	9.3429	1.1072	3.0566	9.3290	1.2574	3.0543
	96	**4.9734**	**0.5439**	**2.2301**	5.3687	0.5804	2.3170	5.0564	0.5722	2.2486	5.0786	0.5774	2.2536	9.5141	1.2329	3.0845	9.3144	1.2135	3.0519	9.3565	1.1552	3.0588

TABLE V
ALBATION STUDY OF MSCNET'S DIFFERENT MODULES ON THREE CLOUD WORKLOAD DATASETS

Models		MSCNet			W/O_Conv			W/O_Patch			W/O_Trend		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	**0.56871**	0.45648	**0.75413**	0.57039	0.45653	0.75524	0.57353	0.45881	0.75732	0.56900	**0.45615**	0.75432
	48	**0.62736**	**0.49863**	**0.79206**	0.62853	0.49861	0.79280	0.63076	0.50072	0.79420	0.62785	0.49888	0.79237
	72	**0.66224**	0.52154	**0.81378**	0.66306	0.52156	0.81429	0.66489	0.52315	0.81541	0.66234	**0.52153**	0.81384
	96	**0.68925**	**0.53789**	**0.83021**	0.69007	0.53823	0.83071	0.69161	0.53950	0.83163	0.68946	0.53857	0.83034
Google	24	**1.03707**	**0.72902**	**1.01837**	1.03754	0.72947	1.01860	1.03801	0.73103	1.01883	1.04263	0.73115	1.02109
	48	1.25548	0.80868	1.12048	**1.25544**	**0.80822**	**1.12046**	1.26063	0.81181	1.12278	1.26050	0.81139	1.12272
	72	**1.41514**	0.86402	**1.18960**	1.41610	0.86473	1.19000	1.41628	0.86423	1.19008	1.41639	**0.86397**	1.19012
	96	**1.52883**	**0.90505**	**1.23646**	1.53397	0.90693	1.23853	1.53124	0.90669	1.23743	1.53745	0.90829	1.23994
Azure	24	**4.06812**	**0.44001**	**2.01696**	4.09376	0.44278	2.02330	4.10397	0.44818	2.02583	4.09127	0.44036	2.02269
	48	**4.31648**	**0.45462**	**2.07761**	4.32701	0.45850	2.08015	4.33394	0.46245	2.08181	4.32244	0.45687	2.07905
	72	**4.44845**	0.47254	**2.10914**	4.45921	**0.47201**	2.11168	4.46484	0.47509	2.11302	4.46365	0.47288	2.11273
	96	**4.55623**	**0.47761**	**2.13453**	4.57456	0.48279	2.13882	4.57301	0.48404	2.13846	4.55897	0.48027	2.13517

sequences from multiple scales, enabling it to adapt to and effectively capture the complex working patterns present in different datasets, thereby demonstrating superior generalization capability and transferability.

C. Ablation Studies

To determine and evaluate the impact of different modules within MSCNet, we conduct ablation studies focusing on the Multi-Scale Convolutions Block, Multi-Scale Patch Block, and Trend Prediction Block. By individually removing each corresponding module, we experimented with several variants of MSCNet across all three datasets. Table V shows the influence of each module on the model's performance. The Multi-Scale Patch Block has the most significant impact on performance, as its omission leads to a decrease in predictive accuracy across all three datasets. This underscores the importance of multi-scale patch learning in capturing various patterns and long-term dependencies in the original cloud workload. The Multi-Scale Convolutions Block and Trend Prediction Block also have notable effects, particularly on the Azure and Google datasets, respectively. The Multi-Scale Convolutions Block can effectively extract local and global features from cloud load data, while the Trend Prediction Block extracts trend information that can help the model achieve more accurate load forecasting. The inclusion of these two modules significantly enhances MSCNet's generalization ability and adaptability to different datasets.

D. Model Analysis

Varing the number of Multi-Scale Block: The key component of MSCNet, the Multi-Scale Block, enables the model to achieve accurate long-term cloud workload forecasting. To assess the impact of the number of Multi-Scale Blocks on prediction accuracy, we conduct experiments with various N, as shown in Table VI. Our findings indicate that in most cases, setting $N = 3$ yields better results. This is because multiple Multi-Scale

TABLE VI
THE PREDICTION ACCURACY VARIES WITH PARAMETER N

		$N=1$			$N=2$			$N=3$		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	0.5720	0.4574	0.7563	0.5693	0.4565	0.7545	**0.5688**	**0.4560**	**0.7542**
	48	0.6294	0.4994	0.7933	0.6282	**0.4986**	0.7926	**0.6279**	0.4987	**0.7924**
	72	0.6635	0.5213	0.8145	0.6627	0.5228	0.8140	**0.6622**	**0.5209**	**0.8138**
	96	0.6904	**0.5375**	0.8309	**0.6897**	0.5379	0.8305	0.6890	0.5376	**0.8301**
Google	24	**1.0373**	0.7301	**1.0185**	1.0377	**0.7294**	1.0187	1.0382	0.7306	1.0189
	48	**1.2547**	**0.8077**	**1.1201**	1.2555	0.8095	1.1205	1.2569	0.8096	1.1211
	72	1.4157	0.8648	1.1898	**1.4148**	**0.8629**	**1.1895**	1.4149	0.8629	1.1895
	96	1.5289	0.9051	1.2365	1.5326	0.9053	1.2380	**1.5285**	**0.9044**	**1.2363**
Azure	24	4.1173	0.4479	2.0291	4.0591	**0.4392**	2.0147	**4.0555**	0.4395	**2.0138**
	48	4.3304	0.4618	2.0810	4.2994	0.4553	2.0735	**4.2964**	**0.4537**	**2.0728**
	72	4.4641	0.4736	2.1128	4.4538	**0.4724**	2.1104	**4.4490**	0.4785	**2.1093**
	96	4.5645	0.4815	2.1365	4.5633	0.4827	2.1362	**4.5417**	**0.4773**	**2.1311**

TABLE VII
THE PREDICTION ACCURACY VARIES WITH PARAMETER S AND P

		$S=1$ $P\in\{16\}$			$S=3$ $P\in\{16,48,96\}$			$S=5$ $P\in\{16,32,48,72,96\}$		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	0.5702	0.4568	0.7551	**0.5686**	0.4563	**0.7541**	0.5688	**0.4560**	0.7542
	48	0.6280	**0.4985**	0.7925	**0.6276**	0.4988	**0.7922**	0.6279	0.4987	0.7924
	72	0.6629	0.5214	0.8142	0.6622	0.5216	0.8138	**0.6622**	**0.5209**	**0.8138**
	96	0.6894	**0.5375**	0.8303	0.6898	0.5388	0.8306	**0.6890**	0.5376	**0.8301**
Google	24	**1.0374**	0.7303	**1.0186**	1.0388	**0.7301**	1.0192	1.0382	0.7306	1.0189
	48	**1.2561**	0.8094	**1.1207**	1.2567	**0.8088**	1.1210	1.2569	0.8096	1.1211
	72	1.4242	0.8683	1.1934	1.4168	0.8638	1.1903	**1.4149**	**0.8629**	**1.1895**
	96	1.5316	0.9066	1.2376	1.5311	0.9056	1.2374	**1.5285**	**0.9044**	**1.2363**
Azure	24	4.0924	0.4455	2.0230	4.0722	0.4424	2.0180	**4.0555**	**0.4395**	**2.0138**
	48	4.3309	0.4606	2.0811	4.3154	0.4588	2.0774	**4.2964**	**0.4537**	**2.0728**
	72	4.4586	0.4734	2.1115	4.4495	**0.4721**	2.1094	**4.4490**	0.4785	**2.1093**
	96	4.5723	0.4832	2.1383	4.5545	0.4800	2.1341	**4.5417**	**0.4773**	**2.1311**

Blocks can more effectively extract and capture the multi-scale features and long-term dependencies of historical cloud workloads. When setting $N=2$ and $N=1$, the model's prediction accuracy shows an acceptable decline, highlighting the powerful capability of our designed Multi-Scale Block. Even a single Multi-Scale Block can achieve relatively accurate long-term cloud workload prediction.

Impact of the number of scale in Multi-Scale Patch Block: MSCNet employs the Multi-Scale Patch Block to extract multi-scale features and different periodic patterns from cloud workloads. To evaluate the impact of different values for S and patch length P on the model's prediction accuracy, we conduct experiments with different S shown in Table VII. Our findings indicate that settings of $S=5$ and $S=3$ yield better results, underscoring the advantage of multi-scale modeling in improving prediction accuracy. As the parameter S decreases, we observe a noticeable decline in performance. Additionally, selecting different patch lengths affects the efficiency of extracting information at various scales, with an appropriate patch length more effectively capturing the different periodic patterns of cloud workloads.

Efficiency analysis: In addition to predictive performance, we also analyze and compare the computational cost and complexity of MSCNet with other baseline methods. Specifically, we compare the number of trainable parameters, MACs (Multiply-Accumulate Operations), average training time per epoch, and inference time per sample. As shown in Table VIII, MSCNet achieves the best performance in terms of training time and inference time benefiting from its well-designed architecture, with MICN yielding similar results. Moreover, we present the

TABLE VIII
NUMBER OF TRAINABLE PARAMETERS, MACS, TRAINING TIME PER EPOCH, AND INFERENCE TIME OF DIFFERENT MODELS ON THE GOOGLE DATASET

Models	Parameters	MACs	Train Time	Infer Time
MSCNet	280.22 K	72.63 M	3.62 s	3.64 ms
TimesNet	18.81 M	11.31 G	13.27 s	14.08 ms
MICN	1.45 M	93.13 M	3.78 s	4.44 ms
FEDformer	17.3 M	791.24 M	33.01 s	46.29 ms
SG-CBA	280.63 K	32.55 M	9.71 s	12.99 ms
esDNN	170.72 K	14.96 M	6.85 s	10.97 ms
L-PAW	192.97 K	19.87 M	5.17 s	7.84 ms

TABLE IX
NUMBER OF TRAINABLE PARAMETERS, MACS, AND COMPLEXITY OF DIFFERENT MSCNET MODULES

Module			Parameters	MACs	Complexity
Trend Prediction Block	Average Pooling		0	12.0 K	$O(L)$
	Linear		2.33 K	232.8 K	$O(LT')$
Multi-Scale Prediction Block	Multi-Scale Block	Multi-Scale Patch Block	23.08 K	7.21 M	$O(L)$
		Transformer Encoder	74.98 K	37.34 M	$O(L^2d)$
		Equidistant Convolution	1.065 K	1.36 M	$O(d)$
		Sparse Convolution	1.065 K	8.62 M	$O(d)$
		Dense Convolution	0.54 K	751.0 K	$O(d)$
		Linear	174.82 K	17.48 M	$O(dL)$
	Linear		2.33 K	232.8 K	$O(LT')$

Parameters, MACs, and theoretical time complexity of different MSCNet modules in Table IX. Overall, the complexity of MSCNet is $O(L^2d)$ due to the use of the Transformer encoder, where L is the length of the input sequence and d is the number of hidden units.

Regarding the number of parameters and MACs, MSCNet is significantly lower than TimesNet, MICN, and FEDformer, but slightly higher than RNN-based workload prediction methods like L-PAW. This is primarily due to MSCNet's more complex model structure, which, given the substantial performance improvement over methods like L-PAW, is an acceptable trade-off. Among all models, TimesNet and FEDformer exhibit the highest computational and time costs due to their more complex encoder-decoder architectures and introduction of frequency-domain transformation.

In summary, MSCNet achieves the highest predictive accuracy compared to other baseline models, while maintaining lower computational cost and complexity, and offering the fastest training and inference time.

V. CONCLUSION

Accurate long-term cloud workload prediction can significantly enhance the efficiency of resource allocation and management in cloud data centers. In this paper, we introduce a novel approach called MSCNet for the long-term prediction of cloud workloads in data centers. MSCNet leverages a Multi-Scale Block, which integrates core modules including the Multi-Scale Patch Block, Transformer Encoder, and Multi-Scale Convolutions Block. This design allows MSCNet to extract and capture multi-scale features, different periodic patterns, and long-term dynamic changes in cloud workloads. MSCNet has been experimentally validated on Alibaba, Google, and Azure datasets.

Compared to other baseline methods, MSCNet demonstrates superior performance.

In the future, our work can be further improved and expanded. We aim to explore more efficient and cost-effective models that leverage adaptive strategies to accommodate the complex and dynamic environment of cloud computing. This includes addressing scenarios such as heterogeneous environments, where resources and workloads may vary significantly across different platforms and architectures. By developing models that can seamlessly integrate and adapt to various types of resources, from traditional servers to cloud-based infrastructures, we can enhance the accuracy and reliability of our predictions.

REFERENCES

[1] K. L. Devi and S. Valli, "Time series-based workload prediction using the statistical hybrid model for the cloud environment," *Computing*, vol. 105, no. 2, pp. 353–374, 2023.

[2] M. T. Islam, H. Wu, S. Karunasekera, and R. Buyya, "SLA-based scheduling of spark jobs in hybrid cloud computing environments," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1117–1132, May 2022.

[3] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, 2017.

[4] Z. Wang, M. M. Hayat, N. Ghani, and K. B. Shaban, "Optimizing cloud-service performance: Efficient resource provisioning via optimal workload allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1689–1702, Jun. 2017.

[5] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[6] S. Li, Y. Wang, X. Qiu, D. Wang, and L. Wang, "A workload prediction-based multi-vm provisioning mechanism in cloud computing," in *Proc. 2013 15th Asia-Pacific Netw. Operations Manage. Symp.*, 2013, pp. 1–6.

[7] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct.-Dec., 2015.

[8] J. Yang et al., "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Inf. Syst. Front.*, vol. 16, pp. 7–18, 2014.

[9] K. Seshadri, K. Sindhu, S. N. Bhattu, and C. Kollengode, "Design and evaluation of a hierarchical characterization and adaptive prediction model for cloud workloads," *IEEE Trans. Cloud Comput.*, vol. 12, no. 2, pp. 712–724, Apr.-Jun., 2024.

[10] D. Saxena, J. Kumar, A. K. Singh, and S. Schmid, "Performance analysis of machine learning centered workload prediction models for cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1313–1330, Apr. 2023.

[11] M. Xu, C. Song, H. Wu, S. S. Gill, K. Ye, and C. Xu, "esDNN: Deep neural network based multivariate workload prediction in cloud computing environments," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–24, 2022.

[12] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 27268–27286.

[13] Y. Liu et al., "iTransformer: Inverted transformers are effective for time series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2024.

[14] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and arima model in cloud data center," in *Proc. 2018 IEEE 15th Int. Conf. Netw., Sens. Control*, 2018, pp. 1–6.

[15] M. Barati and S. Sharifian, "A hybrid heuristic-based tuned support vector regression model for cloud load prediction," *J. Supercomputing*, vol. 71, pp. 4235–4259, 2015.

[16] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *Proc. 2015 IEEE/ACM 10th Int. Symp. Softw. Eng. Adaptive Self- Manag. Syst.*, 2015, pp. 35–45.

[17] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine," *Appl. Intell.*, vol. 48, pp. 4072–4083, 2018.

[18] N. Singh and S. Rao, "Ensemble learning for large-scale workload prediction," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 149–165, Jun. 2014.

[19] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1848–1863, Jul.-Sep. 2022.

[20] Y. Li, H. Hu, Y. Wen, and J. Zhang, "Learning-based power prediction for data centre operations via deep neural networks," in *Proc. 5th Int. Workshop Energy Efficient Data Centres*, 2016, pp. 1–10.

[21] S. Tuli, S. S. Gill, P. Garraghan, R. Buyya, G. Casale, and N. Jennings, "START: Straggler prediction and mitigation for cloud computing environments using encoder LSTM networks," *IEEE Trans. Serv. Comput.*, vol. 16, no. 1, pp. 615–627, Jan.-Feb., 2023.

[22] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A quantum approach towards the adaptive prediction of cloud workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 12, pp. 2893–2905, Dec. 2021.

[23] L. Chen, W. Zhang, and H. Ye, "Accurate workload prediction for edge data centers: Savitzky-golay filter, CNN and bilstm with attention mechanism," *Appl. Intell.*, vol. 52, no. 11, pp. 13027–13042, 2022.

[24] A. I. Maiyza, N. O. Korany, K. Banawan, H. A. Hassan, and W. M. Sheta, "Vtgan: Hybrid generative adversarial networks for cloud workload prediction," *J. Cloud Comput.*, vol. 12, no. 1, 2023, Art. no. 97.

[25] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 95–104.

[26] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang, "SegRNN: Segment recurrent neural network for long-term time series forecasting," 2023, *arXiv:2308.11200*.

[27] Y. Jia, Y. Lin, X. Hao, Y. Lin, S. Guo, and H. Wan, "Witran: Water-wave information transmission and recurrent acceleration network for long-range time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 12389–12456.

[28] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2023.

[29] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," in *Proc. Int. Conf. Mach. Learn.*, 2023.

[30] M. Liu et al., "Scinet: Time series modeling and forecasting with sample convolution and interaction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 5816–5828.

[31] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11121–11128.

[32] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, and J. Kalagnanam, "Tsmixer: Lightweight MLP-mixer model for multivariate time series forecasting," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 459–469.

[33] A. Das, W. Kong, A. Leach, S. K. Mathur, R. Sen, and R. Yu, "Long-term forecasting with tide: Time-series dense encoder," *Trans. Mach. Learn. Res.*, 2023.

[34] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11106–11115.

[35] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 22419–22430.

[36] S. L. Ho and M. Xie, "The use of arima models for reliability forecasting and analysis," *Comput. Ind. Eng.*, vol. 35, no. 1/2, pp. 213–216, 1998.

[37] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv: 1803.01271*.

[38] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "Nhits: Neural hierarchical interpolation for time series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 6989–6997.

[39] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *Proc. Int. Conf. Learn. Representations*, 2022.

[40] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. Int. Conf. Learn. Representations*, 2023.

[41] L. Han, H.-J. Ye, and D.-C. Zhan, "The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7129–7142, Nov. 2024.

[42] J. Bi, S. Li, H. Yuan, Z. Zhao, and H. Liu, "Deep neural networks for predicting task time series in cloud computing systems," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control*, 2019, pp. 86–91.
[43] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *Proc. Int. Symp. Qual. Serv.*, 2019, pp. 1–10.
[44] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format schema," *Google Inc.*, White Paper, vol. 1, pp. 1–14, 2011.
[45] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 153–167.

Feiyu Zhao received the BS degree in automation from the South China University of Technology, in 2022. He is currently working toward the MS degree with the School of Computer Science and Engineering, South China University of Technology, China. His research interests include cloud computing, deep learning, and resource optimization.

Weiwei Lin (Senior Member, IEEE) received the BS and MS degrees from Nanchang University, in 2001 and 2004, respectively, and the PhD degree in computer application from South China University of Technology, in 2007. He has been a visiting scholar with Clemson University from 2016 to 2017. Currently, he is a professor in the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, and AI application technologies. He has published more than 150 papers in refereed journals and conference proceedings. He has been a reviewer for many international journals, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Cybernetics*, etc. He is a distinguished member of CCF.

Shengsheng Lin received the bachelor's degree from the South China University of Technology, in 2022. He is currently working toward the doctoral degree in computer technology with the School of Computer Science and Engineering, South China University of Technology, China. His research interests include machine learning and time series forecasting.

Shaomin Tang received the PhD degree in mechanical engineering from the South China University of Technology, China, in 2022. Currently, she is an associate professor in the School of Electronics and Information Engineering, South China Normal University. Her research interests include AI application in PHM of rotating machinery and computer vision in mechanical inspection.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985 and the PhD degree in computer science from the University of Houston, in 1990. He is a SUNY distinguished professor with the State University of New York and a National Distinguished Professor with Hunan University (China). He has authored or co-authored more than 1080 journal articles, book chapters, and refereed conference papers, and holds more than 75 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has ranked among the world's top few most influential scientists in parallel and distributed computing based on the Scopus citation database (ranked #2 for single-year impact and #4 for career-long impact). He is listed in Scilit Top Cited Scholars (2023 - 2024). He is a member of the SUNY Distinguished Academy, and a fellow of AAAS, IEEE, AAIA, ACIS, and AIIA. He is also a member of the European Academy of Sciences and Arts and Academia Europaea.