# TFEGRU: Time-Frequency Enhanced Gated Recurrent Unit With Attention for Cloud Workload Prediction

Feiyu Zhao, Weiwei Lin, *Senior Member, IEEE*, Shengsheng Lin, Haocheng Zhong, and Keqin Li, *Fellow, IEEE*

***Abstract*—Accurate prediction of cloud workload is crucial for effective resource allocation in cloud computing. However, due to the complexity and high dimensionality of workloads in the cloud environment, achieving precise workload prediction is a complex and challenging problem. Current approaches to cloud workload prediction mainly rely on deep learning methods based on the Recurrent Neural Network (RNN), which struggle to capture the long-term dependencies inherent in workloads effectively. To tackle these challenges and overcome the limitations of existing methods, we propose an effective approach Time-Frequency Enhanced Gated Recurrent Unit with Attention (TFEGRU) for cloud workload prediction. First, we design a Time-Frequency Enhanced Block (TFEB) to capture complex workload patterns and extract features from both the frequency and temporal domains. Next, we integrate channel independent strategy and channel embedding into the model to adapt to high-dimensional workloads and enhance predictive performance. Finally, we apply a Gated Recurrent Unit (GRU) in conjunction with a multi-head self-attention mechanism to achieve accurate workload prediction. To validate the effectiveness of TFEGRU, comprehensive experiments are conducted using real-world traces from Google and Alibaba cloud data centers. The experimental results demonstrate that TFEGRU achieves accurate and efficient predictions across diverse cloud workloads, outperforming existing state-of-the-art methods.**

***Index Terms*—Cloud computing, gate recurrent unit, resource allocation, workload prediction.**

## I. Introduction

In the past decades, cloud computing has gradually become one of the most popular computing paradigms and gained much popularity in the industry [1][2]. Many services, including data storage, social computing, application hosting, and Big Data computing, have been implemented on cloud infrastructure [3][4] [5]. Cloud computing, facilitated by Cloud Service Providers (CSPs), offers users on-demand access to computing, storage, and networking resources while maintaining compliance with Service Level Agreements (SLAs) to guarantee Quality of Service (QoS). Compared to traditional local computing, cloud computing centralizes resource management in data centers, allowing users to access resources anytime via the internet. For instance, Infrastructure as a Service (IaaS) clouds provide tenants with on-demand VM instances [6].

CSPs typically provision resources before the arrival of user requests to ensure QoS. When a large number of user requests arrive simultaneously, the occurrence of bursty workloads may lead to insufficient available resources. Conversely, during periods of low workload, idle states can lead to resource wastage. Imbalances in resource provisioning due to workload variations can result in unnecessary resource expenses or SLA violations. Therefore, CSPs need the capability to promptly identify resource provisioning strategies, ensuring compliance with SLA requirements while optimizing resource utilization [7]. In order to accomplish these goals, cloud computing necessitates precise and efficient methods for predicting workloads. Through effective prediction of future workloads, CSPs can proactively configure and allocate resources in advance. This not only optimizes resource provisioning for greater efficiency and rationality but also helps in energy conservation and emission reduction [8] [9].

Over the past few years, significant research efforts have been directed toward predicting cloud workloads. Traditional approaches for workload prediction predominantly depend on regression methods, machine learning techniques, and deep learning methods. Conventional regression methods like Autoregressive (AR), Moving Average (MA), and Autoregressive Integrated Moving Average (ARIMA) typically rely on linear assumptions, requiring time series stationarity, and struggle with complex nonlinear relationships for high accuracy. As machine learning advances, it's widely applied to cloud workload prediction using methods like random forests, decision trees, and Support Vector Machines (SVM). These models offer a significant advantage over traditional methods in terms of nonlinear

Received 14 April 2024; revised 9 October 2024; accepted 10 November 2024. Date of publication 13 December 2024; date of current version 6 February 2025. This work was supported in part by the National Natural Science Foundation of China under 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, Shengsheng Lin, and Haocheng Zhong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: zhaofy001@foxmail.com; linss2000@foxmail.com; cshczhong@mail.scut.edu.cn).

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).

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.2024.3517324

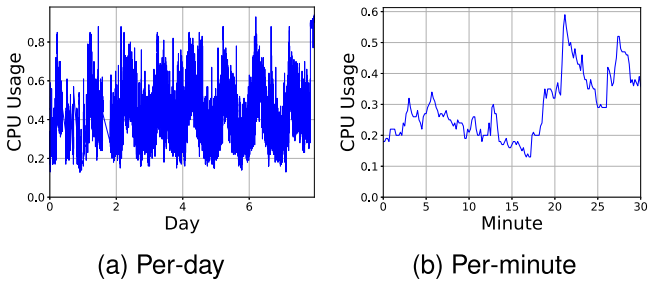


Fig. 1. The random workload in Alibaba cloud data centers.

generalization, enabling them to capture potential relationships between inputs and outputs.

The latest research has commonly utilized deep learning methods based on the Recurrent Neural Network (RNN) and some of its variants [10]. Leveraging the powerful time series modeling capabilities of RNN, these methods outperform regression and machine learning-based approaches, as they can better learn patterns and correlations between workload sequences, exhibiting higher prediction accuracy and learning efficiency.

However, accurate prediction of cloud computing workloads is a complex and challenging problem [11], primarily facing the following two challenges:

- *High Complexity of Workload Patterns.* Workload patterns in cloud computing exhibit varying characteristics at different levels of time scales, as observed in realistic workload traces. For instance, the Alibaba cloud data center analysis report reveals that the average CPU utilization of a cluster can fluctuate between 5% and 80% under significant variations [12], as shown in Fig. 1. The phenomenon of high variance and data distribution drift [13] caused by non-stationary workload data makes accurate and efficient prediction of cloud workloads extremely challenging. Traditional deep learning methods are often influenced by the high complexity of workload patterns, struggling to effectively learn relevant patterns and extract correlations between these patterns from history workload [10].
- *High Dimensionality of Workload Data.* Cloud computing often involves the construction of clusters with a large number of servers, leading to high-dimensional workload data challenges [14]. High-dimensional data introduces noise and redundant information, not only impacting the predictive performance of models but also demanding higher capabilities for feature extraction and learning. Traditional deep learning methods typically model all channels together to learn the correlations between them. However, due to the inherent noise present in cloud workload high-dimensional data, this channel dependent modeling strategy may negatively impact predictive performance.

To address the challenges, we propose a Time-Frequency Enhanced Gated Recurrent Unit with Attention (TFEGRU) approach for accurate, efficient, and robust workload prediction. First, to better capture highly complex cloud workload patterns, we design a Time-Frequency Enhanced Block (TFEB), which integrates both a Frequency Enhanced Module and a Time Enhanced Module. In the Frequency Enhanced Module, we utilize Short-Time Fourier Transform (STFT) to convert temporal information into frequency-domain data, effectively capturing the periodicity of the workload [15]. Meanwhile, the Time Enhanced Module employs convolutional layers to dynamically identify and track changes in workload behavior over time. This allows the model to better detect subtle temporal variations, leading to more accurate extraction of operational patterns in rapidly fluctuating cloud environments. Second, we explore and analyze the influence of channel correlations among different channels of high-dimensional workloads. Our core approach involves modeling each channel independently through channel independent strategy and channel embedding, which eliminates noise interference between high-dimensional data channels, ultimately enhancing the model's adaptability and predictive performance. Finally, we utilize a Gated Recurrent Unit (GRU) in conjunction with a multi-head self-attention mechanism to make accurate workload predictions.

The main contributions of this paper are summarized as follows:

- We propose the Time-Frequency Enhanced GRU with Attention for workload prediction, utilizing the Time-Frequency Enhanced Block to combine both frequency and time-domain information, which enables more effective identification and extraction of complex cloud workload patterns.
- We explore and analyze the influence of channel correlations in high-dimensional workloads, introducing a channel independent strategy and channel embedding to eliminate noise interference from high-dimensional data and enhance predictive performance.
- Extensive experiments utilizing the real-world workload datasets are conducted to validate our approach. The results demonstrate that the proposed method can achieve more accurate workload prediction, outperforming existing approaches.

The remainder of this paper is organized as follows. Section II analyzes related work on workload prediction. Section III provides a detailed description of the model. In Section IV, we evaluate the proposed method through experiments on real-world workload datasets. Finally, we summarize this paper in Section V.

## II. Related Work

Workload prediction in cloud computing has garnered widespread attention in industry and academia. Many studies have been conducted that make notable contributions to this issue. In this section, we first review regression-based workload prediction methods and then introduce machine learning and deep learning methods for workload prediction.

### A. Regression Methods for Workload Prediction

Calheiros et al. [16] proposed a cloud workload prediction method using the ARIMA model, achieving high accuracy and optimal resource utilization. Yang et al. [17] developed an

auto-scaling mechanism based on a linear regression model, ensuring SLA compliance and minimizing costs. Liu et al. [18] introduced an adaptive approach that classifies workloads and assigns them to appropriate prediction models. Bi et al. [19] combined Savitzky-Golay filtering, wavelet decomposition, and ARIMA for better prediction performance than traditional models. Mazumdar et al. [20] enhanced ARMA with wavelet decomposition and Kalman filtering, improving accuracy through seasonal adjustments. Bi et al. [21] introduced a novel approach combining wavelet decomposition, Savitzky-Golay filter, and ARIMA to enhance workload prediction accuracy.

In general, traditional regression methods like AR and ARIMA have proven effective in workload prediction, particularly for workloads with clear patterns or trends. However, these approaches face challenges in efficiently predicting highly variable workloads in the cloud environment [22]. To address these issues, advanced methods such as machine learning and deep learning techniques have been explored.

### B. Machine Learning Methods for Workload Prediction

Tong et al. [23] applied a feature periodic coefficient with various classification techniques, improving success rates and reducing mean square error. Barati et al. [24] proposed TSVR, combining genetic and particle swarm optimization with chaotic sequences to train SVM, enhancing prediction accuracy and preventing premature convergence. Nikravesh et al. [25] used SVM and neural networks, showing that SVM effectively predicted periodic workloads. Cetinski et al. [26] developed a hybrid model integrating classification and regression, significantly improving workload prediction. Zhong et al. [27] introduced a weighted wavelet SVM approach, boosting accuracy by using wavelet functions as SVM kernels and weighting samples.

Overall, these machine learning methods represent stronger workload characteristics, learning ability, and higher performance for workload prediction. However, the majority of machine learning methods perform workload prediction within small-scale grid systems, displaying lower variance when contrasted with cloud data centers [9]. The presence of long-range dependence on cloud workloads makes these methods hard to predict accurately. Thus, more and more scholars are adopting deep learning methods with competent sequence processing ability for cloud workload prediction.

### C. Deep Learning Methods for Workload Prediction

Chen et al. [9] developed a deep learning-based cloud workload prediction method (L-PAW). It utilized sparse autoencoders to extract fundamental representations of workloads, using GRU to learn long-term dependencies and enhance workload prediction accuracy. Bi et al. [28] proposed a workload and resource prediction method for cloud computing, using logarithmic transformation, noise filtering, and Min-Max scaling. By integrating bi-directional and grid LSTM models, the method improves prediction accuracy, outperforming existing approaches on Google cluster trace data. Dogani et al. [29] presented a hybrid model combining Bidirectional Gated-Recurrent Unit (BiGRU), Discrete Wavelet Transformation (DWT), and an attention mechanism to enhance host workload prediction in cloud computing. DWT extracts patterns from nonlinear data, while BiGRU predicts future workloads, leveraging attention to capture temporal correlations. Bi et al. [30] proposed VAMBiG, a hybrid model for predicting cloud workload and resource usage. VAMBiG combines variational mode decomposition, an adaptive Savitzky-Golay filter, multi-head attention, and bidirectional and grid LSTM networks to handle noise, capture nonlinear features, and improve prediction accuracy. Experiments on Google and Alibaba datasets show it outperforms existing methods. Seshadri et al. [31] introduced the Super Markov Prediction Model (SMPM) for workload characterization and prediction in Cloud Data Centers. SMPM adapts its behavior to changing workload patterns over time, employing different sequence models for future workload predictions. Li et al. [32] proposed an Evolution Graph for Workload Prediction (EvoGWP), a method for predicting long-term dynamic workload changes using a graph-based evolution learning algorithm. EvoGWP extracts shapelets to identify fine-grained resource usage patterns, employing a two-level extraction mechanism and an evolution graph model to capture temporal changes and spatial interference among workloads.

In summary, most deep learning methods are improved on the basis of RNN or RNN variant models, which can better learn long-term memory dependencies from the historical workload. Some work has also attempted to use Graph Neural Network (GNN) to predict workload changes from both temporal and spatial dimensions, achieving quite good results [32][33] [34]. However, most existing deep learning methods focus on processing and utilizing time-domain information, lacking the utilization of frequency-domain information, which can more easily learn the long-term dependencies and periodicity in cloud workload. Although few methods are proposed to address the high dimensionality of workload data, such as eliminating noise interference through filters [35] or achieving dimensionality reduction through sparse autoencoders [9], little or no methods attempt to individually model each channel to eliminate noise and interference between different channels for high-dimensional cloud workload prediction.

In order to address these challenges, we first design a Time-Frequency Enhanced Block to identify and capture complex workload patterns from both the time and frequency domains. Next, to mitigate the negative impact of noise across various channels in high-dimensional cloud workloads on model prediction performance, we introduce a channel independent strategy and channel embedding into our model. Finally, we combine GRU with a multi-head self-attention mechanism to achieve accurate predictions of future workloads.

## III. Model

### A. Model Framework

*Preliminary:* Cloud workload prediction task is to predict future workload data using historical workload data. Specifically, given the historical data $X \in R^{L \times D}$ with $L$ length look-back window and $D$ channels, the task is to predict the future workload data $Y \in R^{H \times D}$, where $H$ is the horizon window.

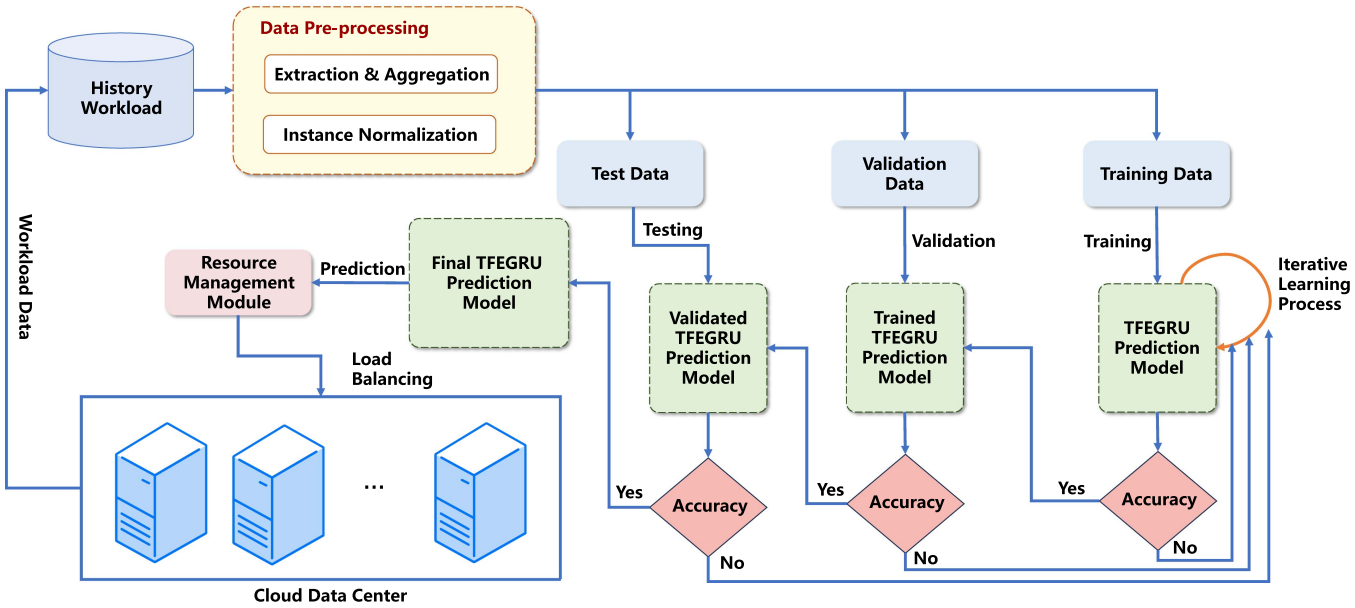


Fig. 2. Workload prediction model in cloud environment.

Accurate predictions of resource information aid in proactively allocating the necessary resources, preventing runtime delays in resource provisioning while meeting QoS constraints [10]. The workflow of the cloud environment workload prediction model is shown in Fig. 2. We preprocess the given historical workload by extracting CPU usage as the primary metric, aggregating data across machines, and normalizing it. The prediction model is then trained and validated on pre-divided datasets until the expected performance is reached. Finally, the trained TFEGRU model predicts future workloads to assist in cloud data center resource management.

*TFEGRU structure:* The overall architecture of TFEGRU is shown in Fig. 4 and composed of tow phases. In the first phase, inspired by the previous works [36][37], we explore a channel independent strategy to process the cloud workload data, modeling each channel of the workload separately. In the second phase, we design the Time-Frequency Enhanced Block (TFEB) to extract features in both the frequency and temporal domains. Then, we utilize a GRU model with a self-attention mechanism to capture long-term dependencies between sequences. Finally, based on the integration of channel embedding and representation learned by GRU, we utilize a linear layer to predict future workload data.

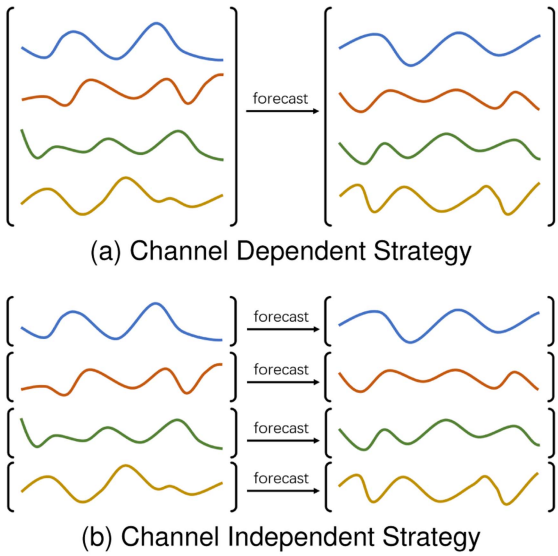


Fig. 3. Illustration of two different strategies for workload prediction. (a) The channel dependent strategy employs the entire set of channels as input, generating forecasts for future workload based on the combined historical data of all channels. (b) The channel independent strategy handles each series individually, training a unified model on these series. In this approach, the prediction for each channel is based on its own historical values.

## B. Data Preprocessing

*Extraction and Aggregation:* The original historical workload data sourced from cloud data centers encompasses a variety of metrics associated with the operational state of the system, including CPU usage, memory usage, disk usage, and I/O time, which undoubtedly adds complexity and redundancy to the computational process. In cloud data centers, the significant cost implications resulting from low CPU utilization are a major concern for CSPs [9]. Meanwhile, the high variance exhibited by CPU utilization in workload performance makes accurate prediction challenging. Therefore, as many previous studies [14][38], we also regard CPU usage as the essential workload performance metric and extract this metric from historical workloads. In order to make the model adapt to the workloads of different machines in cloud data centers, we aggregate different machines' workloads over time to obtain the input $X \in R^{L \times D}$.

*Instance Normalization:* The high variance in workload patterns and the distribution drift of workload data make accurate prediction using traditional methods challenging. To address these issues, we employ a simple instance normalization to alleviate these effects. We begin by subtracting the last value

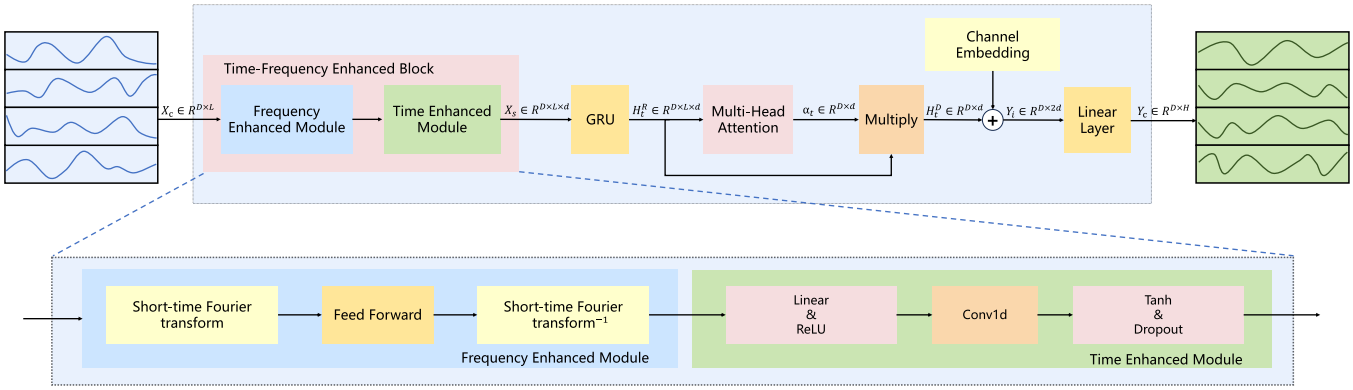


Fig. 4. TFEGRU Prediction Model.

extracted from the input sequence, and finally, we add it back during the output phase as follows:

$$\begin{aligned} x_{1:L}^{(i)} &= x_{1:L}^{(i)} - x_L^{(i)}, \\ y_{L+1:L+H}^{(i)} &= y_{L+1:L+H}^{(i)} + x_L^{(i)}, \end{aligned} \tag{1}$$

where $x_L^{(i)}$ represents the $i$-th channel of workload data at time $L$.

*Channel Independent Strategy:* In many previous studies on workload prediction, methods often overlooked the dependencies among different channels of workload. They typically employed channel dependent approaches (Channel Dependent Strategy, Fig. 3(a)), merging different sequences into hidden layer representations through linear transformations. These models focused on the dependencies among different channels but neglected the negative impact of noise existing between the channels and the disruption of the temporal dependencies within each channel. Recent research [37][39] has shown that modeling channel independence, specifically focusing on the dependencies among different channels, can effectively improve prediction accuracy and enhance the model's robustness against distribution drift. Inspired by these findings, we design and adapt the channel independent approach (Channel Independent Strategy, Fig. 3(b)) for workload prediction, better capturing temporal dependencies among different channels, and improving the model's robustness and performance. For the input series $X \in R^{L\times D}$, we can obtain $X_c \in R^{D\times L}$ adopting the channel independent strategy.

### C. Prediction Model

As shown in Fig. 4, we propose the TFEGRU for accurate cloud workload prediction. First, we design a Time-Frequency Enhanced Block to extract historical workload features from both the frequency and time domains. Subsequently, we employ a GRU with a multi-head self-attention mechanism to learn long-term dependencies between sequences. Finally, combining channel embedding with the representations learned by GRU, we utilize a linear layer to predict future workload. We will introduce more details in the following text.

*Time-Frequency Enhanced Block:* In order to better extract workload features and patterns, we design the Time-Frequency Enhanced Block as shown in Fig. 4, which is composed of Frequency Enhanced Module and Time Enhanced Module.

Within the Frequency Enhanced Module, we employ the classical Short-time Fourier Transform (STFT) method to analyze the temporal frequency characteristics of workload data [15]. Subsequently, we utilize a fully connected layer to capture inherent patterns in the frequency domain and aggregate the frequency information. For the input series $X_c$, the entire process is:

$$\begin{aligned} \widetilde{X} &= \text{STFT}\left(X_c\right), \\ \widetilde{Z} &= \text{Feed Forward}\left(\widetilde{X}\right), \\ Z &= \text{STFT}^{-1}\left(\widetilde{Z}\right). \end{aligned} \tag{2}$$

Unlike the regular Fourier Transform, STFT allows us to observe how the frequency content of a signal changes over short, overlapping time intervals. The STFT process is formulated as follows:

$$\widetilde{X} = \sum_{k=0}^{W-1} \text{Window}[k] \cdot X_t[k + m \times l] \cdot e^{-j\frac{2\pi\cdot\omega\cdot k}{W}}, \tag{3}$$

where $m$ is the index of the sliding window, $\omega$ denotes the frequency, $W$ represents the window length, and $k$ ranges from 0 to $W-1$ as the index of the data in the sliding window and a uniform window with a value of 1 is assumed. Through STFT, we can obtain the time-frequency matrix $\widetilde{X} \in R^{D\times M\times N}$ of the input $X_c$, where $M = \frac{W}{2} + 1$ represents the number of frequency samples and $N = 1 + \frac{L}{l}$ denotes the size of time frames. $l$ is the stride of the moving window in the process of STFT.

Then, we devise the Time Enhanced Module to enhance the extraction of workload features. First, we apply a linear layer with the ReLU activation function to embed the input sequence $Z$. Subsequently, we adopt one-dimensional convolution with the Tanh activation function to capture intricate features of workload. The process is defined as follows:

$$X^{emb} = \text{ReLU}\left(\text{Linear}(Z)\right),$$

$$X_s = \text{Tanh}\left(\text{Conv1d}\left(X^{emb}\right)_{\text{kernel}=1}\right), \tag{4}$$

where $X^{emb}, X_s \in R^{D\times L\times d}$ and $d$ is the hidden state of the series.

*GRU with Attention:* Gated Recurrent Unit (GRU) is a variant of RNN, which employs gate structures to read and update previous information selectively. GRU can effectively address the issue of gradient vanishing, presenting an improvement over traditional RNN. Therefore, we adopt GRU to capture the temporal dependencies in the series. The computing process is as follows:

$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right), \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right), \\ \tilde{h}_t &= \text{Tanh}\left(W \cdot [r_t \odot h_{t-1}, x_t]\right), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \end{aligned} \tag{5}$$

where $z_t$ is the update gate controlling the mix of the previous hidden state $h_{t-1}$ and the candidate hidden state $\tilde{h}_t$, while $r_t$ is the reset gate and $h_t$ is the final state combining $h_{t-1}$ and $\tilde{h}_t$, modulated by $z_t$.

Recent works [40][41] have demonstrated the remarkable capability of the attention mechanism for modeling long-range dependencies. Consequently, we integrate the GRU with multi-head attention to capture the long-term dependencies within time series. To elaborate, we use the GRU output as the input for the multi-head attention, better modeling the dependency relationships across diverse time series. The entire process is defined as follows:

$$\begin{aligned} \alpha_t &= \text{Multi-head Attention}\left(H_t^R, h_{t-1}^R\right), \\ H_t^D &= \alpha_t \odot h_{t-1}^R, \end{aligned} \tag{6}$$

where $H_t^R = [h_{t-L}^R, \ldots, h_{t-1}^R]$ is a matrix stacking the hidden representation of GRU column-wisely. The multi-head attention is a mechanism in deep learning models commonly applied to sequence data. It divides the input sequence into multiple heads, allowing the model to focus on different parts simultaneously. Through the attention mechanism, each head determines the significance of each position in the sequence, enabling the model to capture long-range dependencies. The outputs of multi-head attention $\alpha_t$ are integrated with the latest hidden representation of GRU $h_{t-1}^R$ to form the final representation $H_t^D$, offering enhanced capability in capturing internal relationships within the input sequence.

*Channel Embedding:* Since our model applies the channel independent strategy, in order to make use of each channel of the sequence with a unified model, we are supposed to inject some information about each channel of the sequence. To this end, we add a channel embedding $X_c^{emb} \in R^{D\times d}$ to the input $H_t^D \in R^{D\times d}$. Finally, based on the combination of channel embedding and representation learned by GRU, we use a linear layer to get the final prediction of future workload. The process is as follows:

$$\begin{aligned} Y_i &= \text{Linear}(\text{Concat}\left(X_c^{emb}, H_t^D\right), \\ Y &= \text{Instance Normalization}^{-1}(Y_i), \end{aligned} \tag{7}$$

**Algorithm 1:** Time-Frequency Enhanced GRU With Attention (TFEGRU).

**Input:** Historical workload data $X \in \mathbb{R}^{L\times D}$
**Output:** Future workload data $Y \in \mathbb{R}^{H\times D}$
**Initialize:** look-back window size $L$, horizon window size $H$, workload dimension $D$, hidden units $d$
$X_i = \text{Instance Normalization}(X)$
$X_c = \text{Channel Independent Strategy}(X_i)$
$\widetilde{X} = \text{STFT}(X_c)$
$\widetilde{Z} = \text{Feed Forward}(\widetilde{X})$
$Z = \text{STFT}^{-1}(\widetilde{Z})$
$X^{\text{emb}} = \text{ReLU}(\text{Linear}(Z))$
$X_s = \text{Tanh}(\text{Conv1d}(X^{\text{emb}})_{\text{kernel}=1})$
**for** each look-back window size $L = 1, 2, \ldots, L$ **do**
  Update the update gate $z_t$
  $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
  Update the rest gate $r_t$
  $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
  Calculate the candidate hidden layer $\tilde{h}_t$
  $\tilde{h}_t = \text{Tanh}(W \cdot [r_t \odot h_{t-1}, x_t])$
  Compute the output gate $h_t$
  $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$
**end for**
$\alpha_t = \text{Multi-head Attention}(H_t^R, h_{t-1}^R)$
$H_t^D = \alpha_t \odot h_{t-1}^R$
$Y_i = \text{Linear}(\text{Concat}(X_c^{\text{emb}}, H_t^D))$
$Y = \text{Instance Normalization}^{-1}(Y_i)$
**return** $Y$

where $Y \in R^{H\times D}$ represents the final prediction. The overall procedure of TFEGRU is shown in Algorithm 1.

## IV. Experiments

In this section, we first present the setup of our experiments, including descriptions of datasets, baseline methods, evaluation metrics, and model configuration. Next, the contrasting experimental results of our model and baseline methods on the datasets are discussed. Furthermore, we carry out an ablation study and model analysis to analyze the impact of the model's intrinsic structures and model configuration. All experiments in this section are implemented in PyTorch and executed on a single NVIDIA 4090 GPU equipped with 24 GB of memory.

### A. Experimental Setup

*Datasets:* To evaluate and analyze the performance of our proposed approach, we conduct extensive experiments on two real-world datasets as follows.

- Alibaba Cluster Traces [42]: The Alibaba Cluster Traces dataset was recorded in 2018, denoted as cluster-trace-v2018. It spans a duration of 8 days and consists of approximately 4,000 machines, with a sampling interval of 10 seconds. The data can be accessed on GitHub.[1]

[1] https://github.com/alibaba/clusterdata

- Google Cluster Traces [43]: The Google Cluster Traces dataset captured operational data over a 29-day period in May 2011, providing insights into the performance of a cluster comprising approximately 12.5k machines, referred to as cluster-2011-2. The data can be accessed on GitHub.[2]

Given the crucial role of CPU usage in cloud data centers, we use CPU usage as the main performance metric and extract it from these datasets. In both datasets, we randomly selected 100 machines as the dataset for model evaluation. Subsequently, we aggregate data from different machines over time. Due to some differences between the two datasets, we set the initial prediction granularity for the Google dataset to 5 minutes and for the Alibaba dataset to 10 seconds. In addition, we divide the datasets into three parts, with 60% as the training set, 20% as the validation set, and 20% as the testing set.

*Baselines:* To verify the effectiveness of our approach, several methods are selected for comparison.

- GRU [44] is a type of RNN architecture designed to address the vanishing gradient problem. It exhibits excellent capabilities in capturing dependencies within sequential data.
- L-PAW [9] is an RNN-based cloud workload prediction method. It utilizes top-sparse auto-encoders (TSA) to extract the representations of workloads effectively and integrates TSA and GRU blocks into RNN for adaptive and accurate prediction.
- esDNN [22] represents an efficient approach to cloud workload prediction through supervised learning-based Deep Neural Networks. This method employs a sliding window technique to transform data into a supervised series and leverages a modified GRU for precise workload prediction.
- SG-CBA [41] is a BiLSTM-based deep learning model designed for accurate cloud workload prediction. It incorporates the Savitzky-Golay filter to smooth workload data, followed by the creation of a deep learning module that integrates CNN and BiLSTM with an attention mechanism for accurate workload prediction.
- EvoGWP [32] is an Evolution Graph method for workload prediction. It uses a graph-based evolution learning algorithm to identify fine-grained resource usage patterns by automatically extracting shapelets. EvoGWP combines temporal and spatial dimensions through a spatio-temporal GNN-based encoder-decoder model, enabling accurate prediction of workload changes over time.

*Metrics:* In evaluating the performance of the approaches, three conventional metrics are taken into consideration: the mean absolute error (MAE), mean squared error (MSE), and mean absolute percentage error (MAPE). These metrics are computed as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|,$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2,$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100\%. \tag{8}$$

Among the three metrics, MAE and MSE depend on the scale, while MAPE is scale-independent and represents the Manhattan distance, euclidean distance, and deviation proportion between the actual and predicted values. A lower value for each metric signifies superior performance.

*Configuration:* Based on previous studies and fine-tuning, we configure the look-back window size to 5, while the horizon window size is set to 1. Our model is trained using the L2 loss function, employing the ADAM optimizer with an initial learning rate of $5 \times 10^{-3}$ and a learning rate decay of 0.9. The batch size is chosen as 128, and the model undergoes training for 100 epochs. Early stopping is applied in the training process, halting after five epochs if there is no observed loss degradation on the validation set.

### B. Experimental Results

We compare the prediction accuracy of our proposed TFEGRU model with other methods across various prediction lengths on real-world datasets. Figs. 5 and 6 demonstrate that TFEGRU outperforms other methods consistently across different prediction lengths. Both EvoGWP, esDNN, and SG-CBA show advantages over the native GRU method, with slightly higher predictive accuracy. However, the L-PAW method, which relies on deriving workload representation through TSA, exhibits lower predictive accuracy compared to GRU. Notably, on the Alibaba dataset, TFEGRU performs significantly better at longer prediction lengths, as shown in Fig. 6(a), highlighting its ability to capture long-term dependencies.

Next, to further evaluate the performance of different methods for workload prediction, we conduct experiments with multiple prediction granularities based on the original data sampling intervals of the datasets. For the Google dataset, prediction granularities are chosen from {5min, 15min, 30min, 45min, 1h}. For the Alibaba dataset, prediction granularities are selected from {10s, 30s, 1min, 5min, 10min}. We utilize three metrics, MSE, MAE, and MAPE to assess the performance of these methods, and the results are presented in Table I.

Our method outperforms other methods across all prediction granularities on both datasets, showing significant improvements. For instance, on the Google dataset, TFEGRU achieves a 45.8% reduction in MSE and a 34.9% reduction in MAE at the 5-minute prediction granularity compared to the best-performing method. Similarly, on the Alibaba dataset, TFEGRU reduces MSE by 9% and MAE by 8.1% at the 10-second prediction granularity. Despite the increasing challenge of accurate workload prediction with longer prediction granularity, our approach demonstrates better performance, maintaining smaller performance declines compared to other methods. For example, at a 10-minute prediction granularity on the Alibaba dataset, we achieve a 59.9% reduction in MSE and a 32.7% reduction in MAE compared to other methods. This underscores the effectiveness of our method in adapting to different workload

[2]https://github.com/google/cluster-data

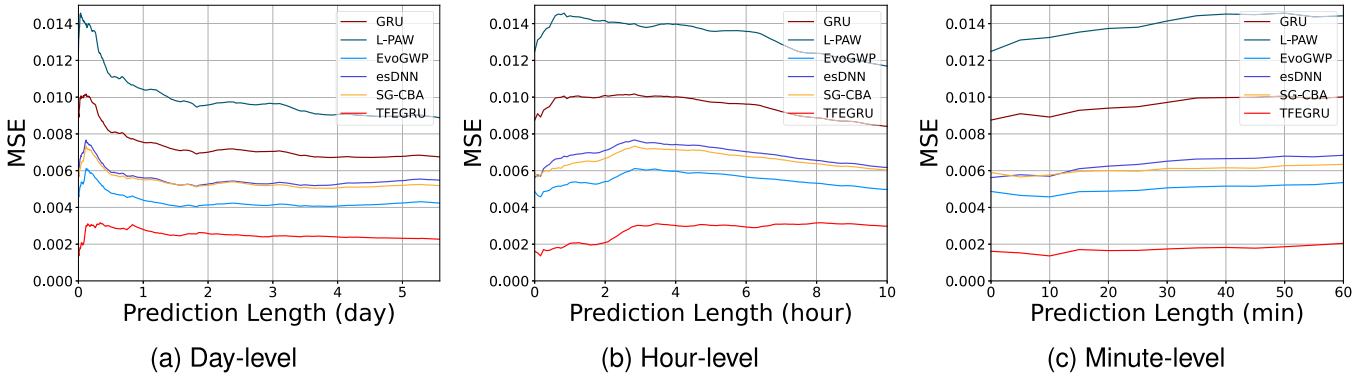


Fig. 5. Prediction accuracy (MSE) of different methods with various prediction lengths on the Google dataset.

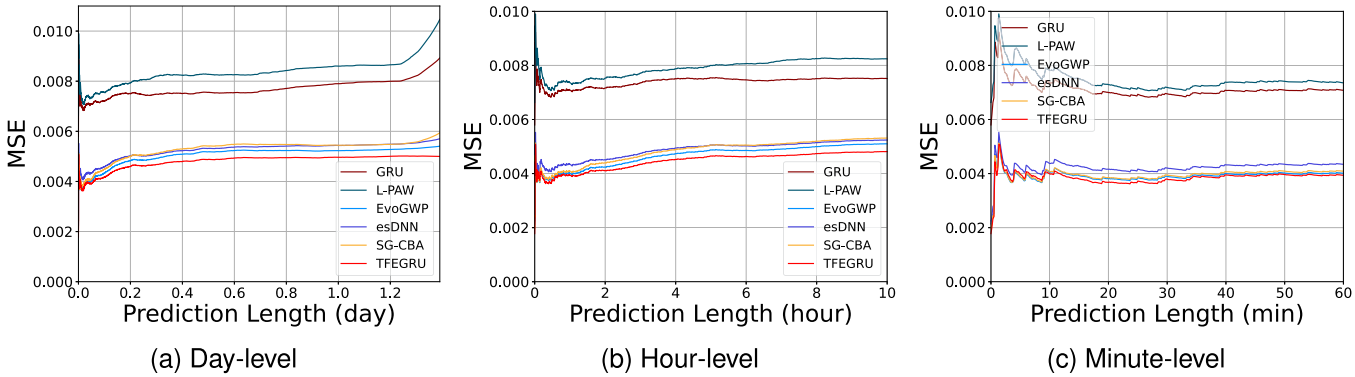


Fig. 6. Prediction accuracy (MSE) of different methods with various prediction lengths on the Alibaba dataset.

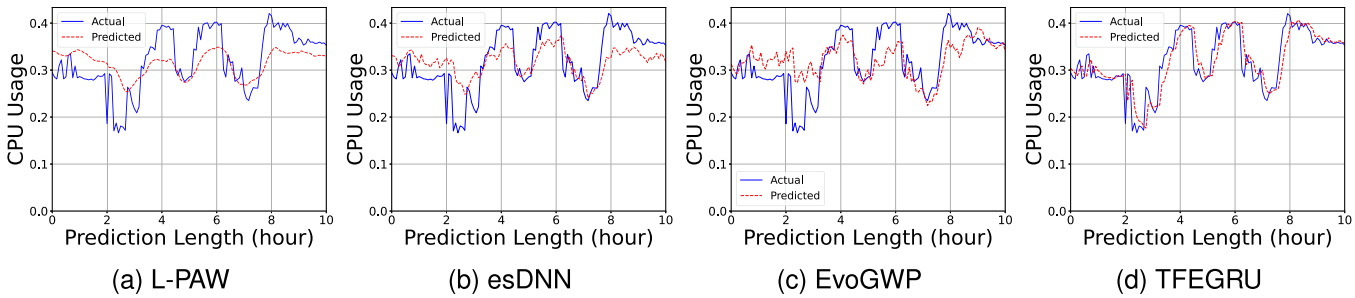


Fig. 7. Prediction performance of different methods with the Google dataset.

patterns, learning long-term dependencies, and achieving higher prediction accuracy and robustness.

Figs. 7 and 8 depict the prediction performance of different methods on the Google and Alibaba datasets. Despite the already high prediction accuracy achieved by the esDNN and EvoGWP methods, TFEGRU exhibits superior predictive accuracy on highly random workloads from Google and Alibaba cloud data centers. The prediction curve provides a better fit to the actual values.

### C. Ablation Studies

To inspect the impact of main components on model performance, we conduct ablation experiments by systematically removing different components, including TFEB, multi-head attention mechanism, channel independent strategy, and channel embedding. Table II summarizes the ablation studies conducted by excluding specific modules from the proposed method. Specifically, the methods for ablation studies are summarized as follows:

- TFEGRU: The complete prediction model.
- TFEGRU-T: Time-Frequency Enhanced Block is excluded from TFEGRU.
- TFEGRU-A: Multi-head attention mechanism is excluded from TFEGRU.
- TFEGRU-C: Channel independent strategy and channel embedding are excluded from TFEGRU.
- TFEGRU-AT: Time-Frequency Enhanced Block and multi-head attention mechanism are excluded from TFEGRU.

TABLE I
MAE, MSE, AND MAPE COMPARISON WITH DIFFERENT PREDICTION GRANULARITY ON THE GOOGLE AND ALIBABA DATASETS

| Dataset | Prediction granularity | Metric | TFEGRU | EvoGWP | SG-CBA | esDNN | L-PAW | GRU |
|---|---|---|---|---|---|---|---|---|
| Google | 5 min | MSE | **0.00226** | 0.00417 | 0.00510 | 0.00608 | 0.00871 | 0.00658 |
| | | MAE | **0.03235** | 0.04966 | 0.05521 | 0.06020 | 0.07237 | 0.06304 |
| | | MAPE | **0.25233** | 0.39116 | 0.41881 | 0.47164 | 0.53014 | 0.47874 |
| | 10 min | MSE | **0.00226** | 0.00678 | 0.00768 | 0.00784 | 0.01045 | 0.00920 |
| | | MAE | **0.03310** | 0.06349 | 0.06677 | 0.06724 | 0.07843 | 0.07393 |
| | | MAPE | **0.18826** | 0.37316 | 0.40454 | 0.40390 | 0.45154 | 0.41248 |
| | 15 min | MSE | **0.00243** | 0.00813 | 0.01038 | 0.00925 | 0.01051 | 0.01126 |
| | | MAE | **0.03399** | 0.06868 | 0.07767 | 0.07411 | 0.07797 | 0.08131 |
| | | MAPE | **0.18212** | 0.37263 | 0.41105 | 0.43997 | 0.40734 | 0.41889 |
| | 30 min | MSE | **0.00256** | 0.00854 | 0.01105 | 0.00884 | 0.01041 | 0.01195 |
| | | MAE | **0.03502** | 0.07021 | 0.08062 | 0.07114 | 0.07797 | 0.08382 |
| | | MAPE | **0.18346** | 0.36791 | 0.40991 | 0.39230 | 0.40163 | 0.40789 |
| | 1 h | MSE | **0.00273** | 0.00891 | 0.00902 | 0.01120 | 0.01120 | 0.01102 |
| | | MAE | **0.03562** | 0.07151 | 0.07213 | 0.08280 | 0.08119 | 0.08012 |
| | | MAPE | **0.18110** | 0.37894 | 0.39840 | 0.47291 | 0.39100 | 0.38680 |
| Alibaba | 10 s | MSE | **0.00495** | 0.00544 | 0.00579 | 0.00575 | 0.01039 | 0.00890 |
| | | MAE | **0.04061** | 0.04417 | 0.04602 | 0.04691 | 0.06945 | 0.06505 |
| | | MAPE | **0.11677** | 0.12940 | 0.13510 | 0.14088 | 0.20443 | 0.20372 |
| | 30 s | MSE | **0.00445** | 0.00532 | 0.00565 | 0.00689 | 0.01364 | 0.00921 |
| | | MAE | **0.04575** | 0.05011 | 0.05192 | 0.05914 | 0.08394 | 0.06989 |
| | | MAPE | **0.13531** | 0.15108 | 0.15826 | 0.18244 | 0.23679 | 0.22376 |
| | 1 min | MSE | **0.00540** | 0.00812 | 0.00921 | 0.01033 | 0.01784 | 0.01185 |
| | | MAE | **0.05135** | 0.06395 | 0.06849 | 0.07313 | 0.09538 | 0.07799 |
| | | MAPE | **0.15068** | 0.19822 | 0.21284 | 0.23045 | 0.26540 | 0.24572 |
| | 5 min | MSE | **0.00703** | 0.01603 | 0.01786 | 0.01740 | 0.02237 | 0.01813 |
| | | MAE | **0.06145** | 0.09045 | 0.09363 | 0.09237 | 0.10367 | 0.09384 |
| | | MAPE | **0.18861** | 0.30489 | 0.32532 | 0.31448 | 0.31978 | 0.32312 |
| | 10 min | MSE | **0.00749** | 0.01868 | 0.02151 | 0.02082 | 0.02668 | 0.02092 |
| | | MAE | **0.06408** | 0.09533 | 0.10133 | 0.09908 | 0.11487 | 0.10010 |
| | | MAPE | **0.21823** | 0.36672 | 0.37756 | 0.37604 | 0.36760 | 0.38332 |

The reported results are averaged over 5 runs. The best results are highlighted in bold, and the second best is underlined.

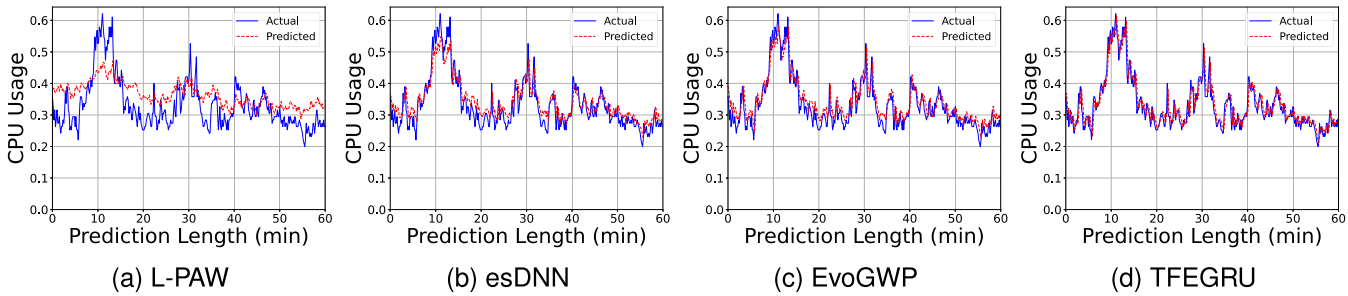


Fig. 8. Prediction performance of different methods with the Alibaba dataset.

- TFEGRU-ATC: Time-Frequency Enhanced Block, multi-head attention mechanism, channel independent strategy, and channel embedding are excluded from TFEGRU.

From Table II, we can observe that TFEGRU exhibits superior performance across most prediction granularities on both datasets. The various TFEGRU variants show a decrease in performance after removing their respective modules, emphasizing the significance of the key modules. TFEGRU-T experiences a more substantial performance drop on the Google dataset, while TFEGRU-A exhibits a larger performance decrease on the Alibaba dataset. Notably, TFEGRU-C demonstrates the most significant performance decline among all variants, consistently yielding the lowest performance at almost each prediction granularity. This further underscores the impact of the channel independent strategy and channel embedding on the high-dimensional cloud workload prediction. Additionally, TFEGRU-AT and TFEGRU-ATC show larger performance declines compared to removing individual modules, indicating that the combination of modules within the model effectively enhances cloud workload prediction performance.

### D. Model Analysis

*Impact of the look-back window and horizon window:* In workload prediction, the look-back window size determines the historical information the model can use. A model with a strong ability to capture long-term temporal dependencies generally performs better as the look-back window increases. Similarly,

TABLE II
ABLATION STUDY OF TFEGRU'S DIFFERENT MODULES ON THE GOOGLE AND ALIBABA DATASETS

| Dataset | Prediction granularity | Metric | TFEGRU | TFEGRU-T | TFEGRU-A | TFEGRU-C | TFEGRU-AT | TFEGRU-ATC |
|---|---|---|---|---|---|---|---|---|
| Google | 5 min | MSE | **0.002260** | 0.002265 | 0.002266 | 0.002559 | 0.002566 | 0.002474 |
| | | MAE | **0.032348** | 0.032376 | 0.032406 | 0.033130 | 0.033183 | 0.033533 |
| | | MAPE | 0.252327 | 0.252144 | 0.251971 | **0.241629** | 0.242511 | 0.249408 |
| | 10 min | MSE | 0.002263 | 0.002275 | **0.002261** | 0.002401 | 0.002434 | 0.002363 |
| | | MAE | 0.033097 | 0.033183 | **0.033064** | 0.033864 | 0.033832 | 0.033958 |
| | | MAPE | 0.188261 | 0.188745 | **0.186158** | 0.189202 | 0.188492 | 0.190166 |
| | 15 min | MSE | **0.002434** | 0.002444 | 0.002434 | 0.002657 | 0.002658 | 0.002507 |
| | | MAE | 0.033985 | **0.033964** | 0.033976 | 0.035146 | 0.035144 | 0.034685 |
| | | MAPE | 0.182121 | 0.182457 | **0.180029** | 0.183979 | 0.183033 | 0.182151 |
| | 30 min | MSE | 0.002565 | 0.002581 | **0.002559** | 0.002788 | 0.002791 | 0.002719 |
| | | MAE | 0.035022 | 0.035049 | **0.034990** | 0.036179 | 0.036224 | 0.036030 |
| | | MAPE | 0.183464 | 0.184530 | **0.182616** | 0.185988 | 0.185548 | 0.184915 |
| | 1 h | MSE | **0.002732** | 0.002773 | 0.002741 | 0.002891 | 0.002896 | 0.002893 |
| | | MAE | **0.035621** | 0.035886 | 0.035745 | 0.036421 | 0.036520 | 0.036591 |
| | | MAPE | **0.181100** | 0.181590 | 0.181814 | 0.182847 | 0.183333 | 0.183965 |
| Alibaba | 10 s | MSE | **0.004953** | 0.004987 | 0.005129 | 0.006717 | 0.006752 | 0.005372 |
| | | MAE | **0.040607** | 0.040819 | 0.041272 | 0.042671 | 0.042836 | 0.041825 |
| | | MAPE | **0.116774** | 0.117095 | 0.117639 | 0.119077 | 0.119888 | 0.120272 |
| | 30 s | MSE | 0.004445 | **0.004439** | 0.004504 | 0.005283 | 0.005300 | 0.004838 |
| | | MAE | **0.045750** | 0.045776 | 0.045940 | 0.047256 | 0.047379 | 0.046860 |
| | | MAPE | **0.135309** | 0.135395 | 0.136102 | 0.139785 | 0.140192 | 0.138921 |
| | 1 min | MSE | 0.005405 | **0.005381** | 0.005500 | 0.006362 | 0.006377 | 0.005905 |
| | | MAE | **0.051349** | 0.051354 | 0.051542 | 0.053369 | 0.053422 | 0.053377 |
| | | MAPE | 0.150678 | **0.150531** | 0.151106 | 0.155565 | 0.155812 | 0.157014 |
| | 5 min | MSE | **0.007030** | 0.007049 | 0.007039 | 0.008064 | 0.008074 | 0.007671 |
| | | MAE | 0.061451 | 0.061659 | **0.061414** | 0.064786 | 0.064843 | 0.064424 |
| | | MAPE | 0.188613 | 0.189082 | **0.187750** | 0.190132 | 0.190202 | 0.195179 |
| | 10 min | MSE | **0.007489** | 0.007515 | 0.007524 | 0.008379 | 0.008383 | 0.007699 |
| | | MAE | **0.064079** | 0.064123 | 0.064307 | 0.067177 | 0.067227 | 0.064881 |
| | | MAPE | 0.218228 | 0.217528 | 0.219031 | 0.215642 | 0.216075 | **0.214350** |

The reported results are averaged over 5 runs. The best results are highlighted in bold, and the second best are underlined.

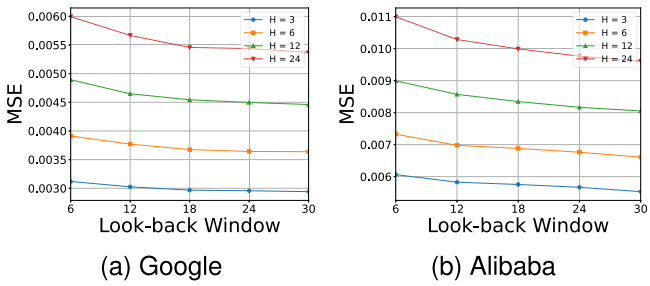


Fig. 9. The prediction error of TFEGRU with different look-back windows and horizon windows on the Google and Alibaba datasets. $H$ represents the horizon window size.

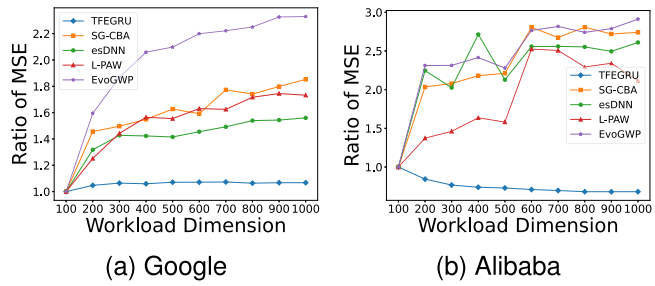


Fig. 10. Comparison of MSE across various workload data dimensions on Google and Alibaba datasets.

the horizon window size is crucial, as a longer horizon implies forecasting further into the future, demanding the model to generalize and maintain accuracy over an extended time span. Therefore, we conduct experiments with different look-back windows and horizon windows to validate our model.

As shown in Fig. 9, as the look-back window size increases, the performance of the model improves across all look-back window configurations. Simultaneously, it is notable that, for a given look-back window size, the model's prediction accuracy decreases with an increase in the horizon window size. However, this decreasing trend gradually diminishes as the look-back window size increases. This observation indicates that TFEGRU can effectively capture long-term temporal dependencies and extract valuable information deeply.

*Impact of workload dimension:* To investigate the impact of workload dimensions on model performance, we conduct experiments across ten different workload dimensions ranging from 100 to 1000 on both the Google and Alibaba datasets. As depicted in Fig. 10, each curve represents the Ratio of MSE for different methods across various workload dimensions, with the values at dimension 100 used as the baseline. It is evident that, with an increase in workload dimensions, the Ratio of MSE escalates rapidly for most methods, while our approach

TABLE III
NUMBER OF TRAINABLE PARAMETERS, MACS, TRAINING TIME, AND PREDICTION TIME OF DIFFERENT MODELS ON THE GOOGLE DATASET

| Model | MACs | Parameters | Training Time | Prediction Time |
|---|---|---|---|---|
| TFEGRU | 5.04G | 189.48K | 689.00ms | 73.00ms |
| TFEGRU-T | 2.93G | 123.01K | 595.05ms | 59.34ms |
| TFEGRU-A | 3.75G | 123.43K | 668.42ms | 66.62ms |
| TFEGRU-C | 85.67M | 201.94K | 360.78ms | 35.34ms |
| esDNN | 87.69M | 151.47K | 170.00ms | 23.00ms |
| L-PAW | 219.46M | 1.28M | 187.00ms | 25.00ms |
| SG-CBA | 177.53M | 283.60K | 207.45ms | 24.92ms |
| GRU | 54.60M | 75.57K | 131.30ms | 19.20ms |
| EvoGWP | 63.99M | 244.65K | 184.43ms | 33.44ms |

maintains stability and even exhibits a slight decrease on the Alibaba dataset.

This observation underscores the effectiveness of our model's channel independent strategy and channel embedding. By individually processing each channel of workload and employing a unified model for learning, our approach demonstrates enhanced robustness and capacity to extract information from high-dimensional workloads.

*Efficiency analysis:* To compare the complexity of different methods, we compare the number of trainable parameters, MACs,[3] training time, and prediction time, as shown in Table III. Compared to other methods, TFEGRU exhibits higher complexity. This is because TFEGRU incorporates modules such as the Time-Frequency Enhanced Block, multi-head attention, channel independent strategy, and channel embedding, which inevitably increase the complexity while improving model performance. Therefore, to better evaluate the efficiency of our method, we also compare several variants of the model. It can be seen that TFEGRU-T and TFEGRU-A have fewer trainable parameters compared to other methods, while TFEGRU-C shows a significant reduction in MACs, training time, and prediction time. Additionally, the variants of TFEGRU outperform other methods in terms of performance. Considering the performance improvement in prediction accuracy, this is an acceptable cost.

## V. CONCLUSION

Accurate and adaptive workload prediction is crucial for efficient resource allocation in cloud computing. However, workload prediction faces challenges arising from the workload's complex patterns and high-dimensional characteristics. In this paper, we propose an accurate and adaptive cloud workload prediction approach TFEGRU. We initially design TFEB to extract complex workload patterns deeply from both the frequency and temporal domains. Subsequently, we explore and integrate channel independent strategy and channel embedding into TFEGRU to enhance the model's robustness and prediction accuracy. Finally, we employ GRU in conjunction with a multi-head self-attention mechanism to achieve adaptive and accurate prediction of highly complex workloads. Extensive experiments using real workload datasets are conducted. The experimental results demonstrate that our proposed method outperforms state-of-the-art approaches in terms of accuracy and adaptability.

Based on the promising results obtained in our current work, our future efforts will focus on exploring longer-term workload prediction. This aims to provide an improved lead time for cloud computing resource allocation and offer valuable insights for proactive decision-making.

[3]MACs (Multiply-Accumulate Operations) refers to the number of multiply and accumulate operations in a neural network model.

## REFERENCES


[1] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," in *Proc. IEEE*, vol. 102, no. 1, pp. 11–31, Jan. 2014.

[2] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 31–42, Jan. 2018.

[3] 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.

[4] F. Liu, B. Luo, and Y. Niu, "Cost-effective service provisioning for hybrid cloud applications," *Mobile Netw. Appl.*, vol. 22, pp. 153–160, 2017.

[5] Y. Li, F. Liu, Q. Chen, Y. Sheng, M. Zhao, and J. Wang, "MarVeLScaler: A multi-view learning-based auto-scaling system for MapReduce," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 506–520, First Quarter, 2022.

[6] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2470–2483, Aug. 2016.

[7] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, Mar. 2016.

[8] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin, "Carbon-aware online control of geo-distributed cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2506–2519, Sep. 2016.

[9] 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.

[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. 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.

[12] W. Chen, K. Ye, Y. Wang, G. Xu, and C.-Z. Xu, "How does the workload look like in production cloud? Analysis and clustering of workloads on alibaba cluster trace," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 102–109.

[13] 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.

[14] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Trans. Ind. Inform.*, vol. 14, no. 7, pp. 3170–3178, Jul. 2018.

[15] B. Boashash, *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. New York, NY, USA: Academic Press, 2015.

[16] 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, Fourth Quarter, 2015.

[17] 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.

[18] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *J. Netw. Comput. Appl.*, vol. 80, pp. 35–44, 2017.

[19] 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. IEEE 15th Int. Conf. Netw., Sens. Control*, 2018, pp. 1–6.

[20] S. Mazumdar and A. S. Kumar, "Forecasting data center resource usage: An experimental comparison with time-series methods," in *Proc. 8th Int. Conf. Soft Comput. Pattern Recognit.*, Springer, 2018, pp. 151–165.

[21] J. Bi, H. Yuan, S. Li, K. Zhang, J. Zhang, and M. Zhou, "ARIMA-based and multiapplication workload prediction with wavelet decomposition and Savitzky–Golay filter in clouds," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 54, no. 4, pp. 2495–2506, Apr. 2024.
[22] 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.
[23] J.-J. Tong, E. Hai-Hong, M. -N. Song, and J.-D. Song, "Host load prediction in cloud based on classification methods," *J. China Universities Posts Telecommun.*, vol. 21, no. 4, pp. 40–46, 2014.
[24] 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.
[25] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *Proc. IEEE/ACM 10th Int. Symp. Softw. Eng. Adaptive Self- Manag. Syst.*, 2015, pp. 35–45.
[26] K. Cetinski and M. B. Juric, "AME-WPC: Advanced model for efficient workload prediction in the cloud," *J. Netw. Comput. Appl.*, vol. 55, pp. 191–201, 2015.
[27] 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.
[28] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, 2021.
[29] J. Dogani, F. Khunjush, and M. Seydali, "Host load prediction in cloud computing with discrete wavelet transformation (DWT) and bidirectional gated recurrent unit (BiGRU) network," *Comput. Commun.*, vol. 198, pp. 157–174, 2023.
[30] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid LSTM for cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375–384, Third Quarter, 2023.
[31] 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, Second Quarter, 2024.
[32] J. Li, J. Yao, D. Xiao, D. Yang, and W. Wu, "EvoGWP: Predicting long-term changes in cloud workloads using deep graph-evolution learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 3, pp. 499–516, Mar. 2024.
[33] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 17804–17815.
[34] Z. Wei et al., "STGSA: A novel spatial-temporal graph synchronous aggregation model for traffic prediction," *IEEE/CAA J. Automatica Sinica*, vol. 10, no. 1, pp. 226–238, Jan. 2023.
[35] 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.
[36] 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.
[37] 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.
[38] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *Proc. 29th Int. Conf. Comput. Commun. Netw.*, 2020, pp. 1–9.
[39] 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.
[40] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "ASRNN: A recurrent neural network with an attention model for sequence labeling," *Knowl.-Based Syst.*, vol. 212, 2021, Art. no. 106548.
[41] 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.
[42] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *Proc. Int. Symp. Qual. Service*, 2019, pp. 1–10.
[43] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format schema," *Google Inc., White Paper*, vol. 1, pp. 1–14, 2011.
[44] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

**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) is a professor with 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*, etc.

**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.

**Haocheng Zhong** received the bachelor's degree from the School of Computer Science and Technology, University of Science and Technology of China, in 2022. He is currently working toward the PhD degree with the School of Computer Science and Engineering, South China University of Technology. His research areas include cloud computing, cluster computing, and cluster performance modeling.

**Keqin Li** (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing, and mobile edge computing. He has published more than 740 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Sustainable Computing*, etc..