The Journal of Supercomputing (2026) 82:461
https://doi.org/10.1007/s11227-026-08598-z

WDBNet: wavelet dual-branch network for long-term cloud workload prediction

Zhenkai Yuan[1] · Bo Liu[1] · Shaomin Tang[4] · Weiwei Lin[2,3] · Keqin Li[5]

Received: 26 November 2025 / Accepted: 9 May 2026

Abstract

Accurate prediction of cloud workload is essential for efficient resource allocation and capacity planning in large-scale distributed and supercomputing environments. While short-term forecasting has made notable progress, long-term prediction remains challenging due to the need to simultaneously model both short-term and long-term dependencies and capture multi-scale features. Moreover, many existing models rely on complex architectures that result in high inference overhead, limiting their practical applicability. To overcome these challenges, we propose the Wavelet Dual-Branch Network (WDBNet), a long-term cloud workload prediction model that integrates wavelet decomposition with a dual-branch channel strategy to balance accuracy and efficiency. Leveraging the observation that low-frequency components exhibit strong cross-channel correlations while high-frequency components behave more independently, WDBNet employs a dual-branch architecture: the long-term branch capturesglobal

✉ Shaomin Tang
tangshaomin@scnu.edu.cn

✉ Weiwei Lin
linww@scut.edu.cn

Zhenkai Yuan
yuanzk001@foxmail.com

Bo Liu
liugubin530@126.com

Keqin Li
lik@newpaltz.edu

1 School of Computer Science, South China Normal University, Guangzhou 510631, Guangdong, China

2 School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, Guangdong, China

3 Pengcheng Laboratory, Shenzhen 518055, Guangdong, China

4 School of Electronics and Information Engineering, South China Normal University, Guangzhou 510631, Guangdong, China

5 Department of Computer Science, State University of New York at New Paltz, New York 12561, USA

Published online: 28 May 2026

trends and inter-channel dependencies, while the short-term branch models local variations within each channel. Extensive experiments on the Alibaba, Azure, Bitbrain, and Google datasets demonstrate that WDBNet improves prediction performance in terms of mean squared error (MSE) by 1.3%, 3.1%, 3.3%, and 2.0%, respectively, compared to state-of-the-art methods, while reducing the number of parameters by 41.3%. These results validate the effectiveness of WDBNet in achieving high predictive accuracy with reduced computational complexity. We also integrated it into auto-scaling scenarios for experimentation. The results demonstrate that WDBNet significantly reduces the number of active machines, thereby improving resource utilization efficiency.

Keywords Cloud computing · Workload prediction · Long-term prediction · Multi-scale decomposition · Dual-branch network

1 Introduction

In recent years, cloud computing has emerged as a disruptive information technology paradigm that is fundamentally reshaping the computing landscape for enterprises, research institutions, and individual users [1]. Today's cloud data centers increasingly serve as the backbone infrastructure for high-performance computing (HPC) and massive distributed applications. As cloud service providers (CSPs) such as Alibaba Cloud and Google Cloud continue to expand their services and support large-scale application deployment through elastic resource provisioning, the operational tasks of cloud platforms have become increasingly complex. Managing resources at such a supercomputing scale requires predictive models to be highly scalable and capable of real-time execution. Consequently, achieving efficient resource scheduling and capacity planning while ensuring quality of service (QoS) compliance with service level agreements (SLAs) has become a critical challenge that urgently needs to be addressed.

Cloud workload, which refers to users' requests for computing resources (e.g., CPU, memory, I/O), exhibits significant temporal dependency, burstiness, and non-stationarity [2]. Traditional static or manual configurations struggle to handle workload fluctuations, often resulting in SLA violations and resource wastage. Consequently, increasing research attention has been directed toward cloud workload prediction techniques to enable proactive resource allocation and achieve an optimal balance between performance and cost.

Despite the application of various regression [3, 4] and machine learning [5] approaches in cloud workload prediction, existing approaches generally focus on short-term forecasting and provide limited support for scenarios requiring advanced resource provisioning. In such cases, long-term prediction not only offers greater strategic value by enabling proactive resource scheduling and capacity planning, but also imposes stricter demands on model performance and efficiency. However, existing approaches, whether tailored for cloud workload prediction or for general long-term time series forecasting, still face two major challenges.[6] The first is limited predictive accuracy. Current models often struggle to capture both local bursty behaviors and long-term trends inherent in cloud workloads, resulting in inadequate modeling of complex, multi-scale temporal dynamics. This limitation primarily stems from

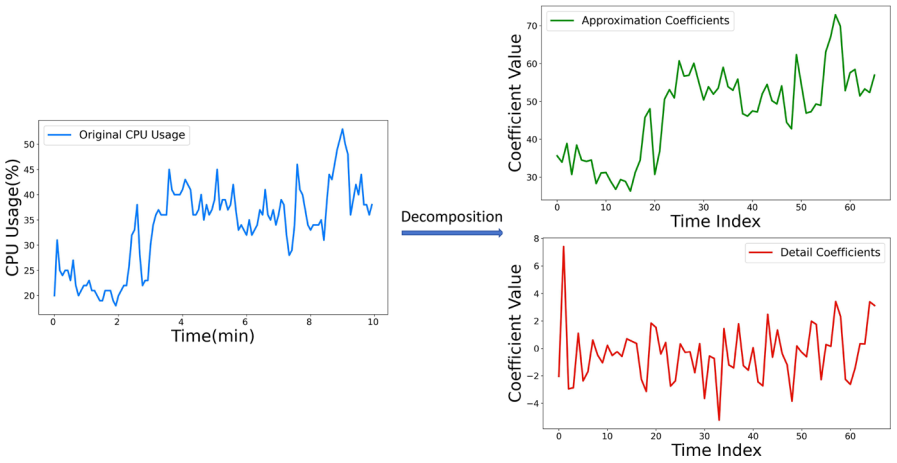


Fig. 1 Wavelet decomposition of real-world cloud workload trace

architectural designs that fail to fully exploit the unique statistical characteristics of cloud workloads. The second challenge is computational inefficiency. Many existing approaches rely on deep and complex neural network architectures with large parameter counts, leading to high training and inference costs. These limitations hinder the practical deployment of such models in real-world cloud systems. Therefore, there is an urgent need for a long-term cloud workload prediction approach that strikes a balance between predictive accuracy and computational efficiency to meet the demands of practical cloud computing environments.

To enhance both prediction accuracy and model efficiency, we explore the inherent characteristics of cloud workload. Cloud workloads typically exhibit pronounced non-stationarity[7] and multi-scale features, characterized by slowly evolving trends at large scales while frequently containing high-frequency fluctuations at small scales.[8] To leverage these characteristics, we introduce wavelet decomposition [9][10] to separate long-term trends from short-term fluctuations in the time-frequency domain, enabling more targeted feature modeling.[11] As shown in Fig. 1, taking a cloud server's CPU utilization sequence as an example, one-level wavelet decomposition produces approximation coefficients representing long-term trends and detail coefficients capturing short-term burst fluctuations. This decomposition provides two key insights. On the one hand, it enhances predictive performance by enabling frequency-specific modeling strategies for global and local patterns. On the other hand, the differentiated and lightweight design opportunities across branches create potential for improved model efficiency.

Furthermore, we observe that different frequency components exhibit distinct dependency patterns along the channel dimension. As shown in Fig. 2, we decompose and compare the approximation and detail coefficients from CPU utilization sequences of two representative servers. Our analysis reveals stronger inter-channel correlations among approximation coefficients, while detail coefficients exhibit weaker dependencies and greater independence across channels. This finding motivates

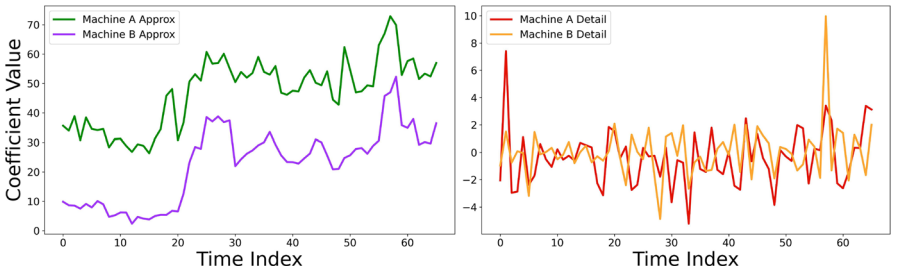


Fig. 2 Comparison of wavelet-decomposed coefficients across different cloud workload sequences

our differentiated channel modeling strategy: for low-frequency components dominated by long-term patterns, we employ the channel-dependent strategy to capture cross-channel collaborative trends, while for high-frequency components, the channel-independent strategy is adopted to enhance both efficiency and prediction performance.

Based on these observations, we propose the Wavelet Dual-Branch Network (WDBNet), an efficient and accurate model for long-term cloud workload prediction. WDBNet first performs multi-level wavelet decomposition on the original sequence to separate low-frequency and high-frequency subsequences, effectively mitigating non-stationarity while explicitly modeling patterns at different scales. The architecture then employs a dual-branch design: the long-term branch processes low-frequency components using the channel-dependent strategy to capture cross-machine long-term collaborative relationships, while the short-term branch handles high-frequency components through the channel-independent strategy to enhance perception of burst dynamics in individual machines. To balance modeling capacity with efficiency, both branches incorporate patch partitioning and depthwise separable convolutions(DSC) for efficient local feature extraction, significantly reducing computational costs while maintaining prediction accuracy. In the subsequent architecture, the long-term branch introduces channel encoders to strengthen global trend modeling, while the short-term branch employs lightweight MLPs for independent per-channel processing, thereby improving rapid response capability to short-term fluctuations.

The key contributions of this paper can be summarized as follows:

- We propose the Wavelet Dual-Branch Network (WDBNet) for long-term cloud workload prediction. This network integrates wavelet decomposition with a dual-branch modeling strategy to separately capture long-term trends and short-term fluctuations, significantly improving long-term forecasting performance.
- We propose an efficient dual-branch modeling framework. Both branches employ patch partitioning and DSC for local feature extraction: the long-term branch leverages the channel-dependent strategy and a channel encoder to capture trends, while the short-term branch combines the channel-independent strategy with an inter-patch MLP to capture rapid fluctuations and reduce redundant computation.
- Extensive experiments on various cloud workload datasets from Alibaba, Google, Azure, and Bitbrain demonstrate that WDBNet achieves superior performance and efficiency compared to state-of-the-art approaches.

The remainder of this paper is organized as follows: Sect. 2 reviews related work. Section 3 presents the WDBNet architecture in detail. Section 4 provides experimental evaluation. Section 5 concludes this paper and discusses future research directions.

2 Related work

In this section, we review related work in both general long-term time series prediction and cloud workload prediction.

2.1 General long-term time series prediction

In recent years, the focus of general time series prediction research has gradually shifted from Short-term Prediction to Long-term Prediction. Compared to Short-term Prediction, Long-term Prediction holds greater value in domains such as weather, energy, transportation, and finance, enabling early estimation of future trends to support planning and decision-making. Existing Long-term Prediction models mainly adopt architectures based on RNNs, CNNs, MLPs, and Transformers, which will be briefly introduced below.

RNN-based models can accumulate temporal contextual information in hidden states, enabling them to capture temporal dependencies, such as SegRNN [12] and SutraNets [13]. However, RNNs suffer from limitations such as long-term memory decay and information bottlenecks when processing long sequences.

CNN-based models efficiently extract local patterns within sequences through local receptive fields, offering high parallelism and computational efficiency. Recent works such as TimesNet [14], MICN [15], and PatchMixer [16] leverage convolutional operations to improve the modeling of periodic patterns in time series. However, CNNs are inherently limited in capturing global temporal dependencies.

MLP-based models exhibit strong nonlinear transformation capabilities and the ability to capture global dependencies. Lightweight models such as DLinear [17] and TSMixer [18] demonstrate excellent forecasting performance with reduced parameter counts. However, when dealing with sequences characterized by non-stationarity (e.g., cloud workload traces), MLP-based approaches face challenges in effectively modeling their complex dynamics.

Transformer-based models leverage self-attention mechanisms to achieve superior long-term dependency modeling capabilities compared to RNNs and CNNs. However, the computational complexity of their attention mechanisms scales quadratically as $O(L^2)$, resulting in significant resource overhead when processing long sequences. To address this, researchers have developed methods like Informer [19] and PatchTST [20], which employ sparse attention and patching mechanisms to reduce computational costs. Meanwhile, FEDformer [21] introduces a frequency-enhanced decomposition architecture to effectively capture global temporal patterns in the frequency domain with linear complexity. Furthermore, enhanced architectures including iTransformer [22], TimeBridge [23], and MultiPatchFormer [24] have been introduced to improve the modeling capacity for complex temporal structures.

2.2 Cloud workload prediction

Existing approaches for cloud workload prediction can be grouped into four major categories: traditional regression, ensemble learning, classical machine learning, and deep learning approaches. A comprehensive taxonomy of these approaches is summarized in Table 1.

Traditional regression methods were widely adopted in early applications due to their simple structure and computational efficiency. Li et al. [3] proposed the ARIMA-DEC model, which reduces SLA violation rates and resource costs through dynamic error compensation. Singh et al. [25] developed the TASM framework by integrating ARIMA, linear regression, and SVR to improve adaptability. Xie et al. [26] combined ARIMA with triple exponential smoothing to optimize container workload prediction. Bi et al. [4] introduced SWARIMA, incorporating Savitzky-Golay filtering and wavelet decomposition to enhance multi-task forecasting capability. However, these methods are only suitable for stationary time series and struggle with non-stationary sequences, leading to their gradual replacement by ensemble learning approaches.

Ensemble learning methods enhance prediction accuracy and robustness through multi-model fusion. Kaur et al. [27] proposed REAP, which employs genetic algorithms to combine multiple regression models for improved CPU utilization prediction accuracy. Kim et al. [28] developed CloudInsight, adopting an online weighting mechanism to dynamically ensemble multiple predictors for handling sudden fluctuations. Although ensemble learning improves performance, it involves higher computational overhead in model construction and deployment. To enhance efficiency, researchers have subsequently shifted toward more flexible machine learning approaches.

Machine learning methods demonstrate superior capability in capturing nonlinear patterns in data. Nikravesh et al. [5] proposed an adaptive prediction system combining SVM and neural networks, which enhances accuracy in multi-workload cases through sliding window and online learning strategies. Barati et al. [29] developed a hybrid SVR-based model incorporating genetic algorithms and particle swarm optimization to improve convergence speed. However, these methods remain constrained by strong feature dependency and limited structural expressiveness when handling multi-scale variations and non-stationary cases.

To address these challenges, deep learning approaches have been widely adopted for complex workload prediction. For instance, Chen et al. [30] proposed L-PAW using sparse autoencoders and GRUs, Xu et al. [31] developed esDNN with enhanced GRUs, and Chen et al. [32] introduced SG-CBA by integrating filtering, CNNs, and BiLSTMs. More recently, Dogani et al. [33] combined discrete wavelet transformation (DWT), attention mechanisms, and BiGRUs for accurate short-term prediction, while Li and Zhang [34] proposed a hybrid architecture integrating Variational Mode Decomposition (VMD), wavelet denoising, TCNs, and GRUs to enhance robustness against noise. However, these methods primarily focus on short-term forecasting and struggle to capture the complex long-range dependencies required for extended horizons. Although [33] and [34] share our frequency-domain perspective, they fundamentally differ from our work in their core objectives. In contrast, our WDBNet is explicitly

tailored for long-term forecasting scenarios, designed to effectively tackle the exceptionally complex global dependencies present in extended horizons.

Recently, several studies have begun to focus on long-term workload prediction tasks. EvoGWP [35] leverages shapelet extraction and graph neural networks to model the evolution of resource usage patterns, while MSCNet [36] employs multi-scale blocks to capture both short-term and long-term dependencies as well as periodic characteristics. However, EvoGWP exhibits limited performance in long-term forecasting tasks. MSCNet, though achieving competitive results across multiple evaluation metrics, suffers from two key limitations: its inference efficiency remains suboptimal in certain scenarios, and its representation of latent workload features remains inadequate. These shortcomings constrain the model's expressive power and generalizability. Furthermore, while existing general long-term time series forecasting models demonstrate strong cross-dataset generalization, they often struggle to effectively learn the deep structures and evolving patterns in highly dynamic, complex, and non-stationary sequences such as cloud workloads–resulting in significantly degraded prediction performance.

To address the above challenges, this paper proposes WDBNet, a deep learning model that integrates wavelet decomposition with a dual-branch channel strategy. Specifically, wavelet decomposition is applied to perform multi-scale analysis of workload sequences, and a dual-branch architecture is designed to separately model different frequency components, enabling differentiated representation of high-frequency fluctuations and low-frequency trends. The model aims to improve both the accuracy and efficiency of long-term cloud workload prediction.

3 Model

3.1 Problem formulation

In long-term cloud workload prediction tasks, we address the following problem: given a set of historical values of cloud workloads $X = (x_1, x_2, ..., x_L) \in \mathbb{R}^{L \times C}$, our goal is to predict their future values $Y = (y_1, y_2, ..., y_T) \in \mathbb{R}^{T \times C}$. Here, L denotes the size of the historical window, T represents the size of the prediction window, and C indicates the number of channels. In our setting, each channel corresponds to one machine.

3.2 Model architecture

The overall architecture of WDBNet is shown in Fig. 3. This model combines multi-level wavelet decomposition with a dual-branch architecture to capture long-term trends and short-term fluctuations of cloud workloads from a multi-scale perspective, thereby achieving efficient and accurate long-term prediction.

First, WDBNet employs multi-level wavelet decomposition to separate the original sequence into approximation and detail coefficients, which correspond to low-frequency trends and high-frequency perturbations, respectively. To process these distinct components, WDBNet utilizes an asymmetric dual-branch architecture.

Table 1 Comparison of existing workload prediction works

Approach	Technique								Performance metrics					
	regression	ensemble	ML	Deep Learning					MSE	MAE	RMSE	MAPE	R2	CDF
				GNN	RNN	CNN	MLP	Transformer						
Li et al. [3]	✓								✓					
TASM [25]	✓								✓	✓	✓	✓		
Xie et al. [26]	✓								✓			✓		
Bi et al. [4]	✓								✓			✓	✓	
Kaur et al. [27]		✓									✓		✓	
CloudInsight [28]		✓									✓	✓		✓
Barati et al. [29]			✓						✓	✓	✓	✓		
Nikravesh et al. [5]			✓								✓	✓		
L-PAW [30]					✓				✓					✓
esDNN [31]					✓	✓			✓		✓	✓		✓
SG-CBA [32]					✓	✓			✓	✓			✓	
DWT-BiGRU-attention [33]					✓				✓	✓	✓	✓		
VNWSTG [34]					✓	✓			✓	✓	✓			
EvoGWP [35]				✓	✓					✓	✓			
MSCNet [36]						✓	✓	✓	✓	✓	✓			
WDBNet(Our Work)						✓	✓	✓	✓	✓	✓			

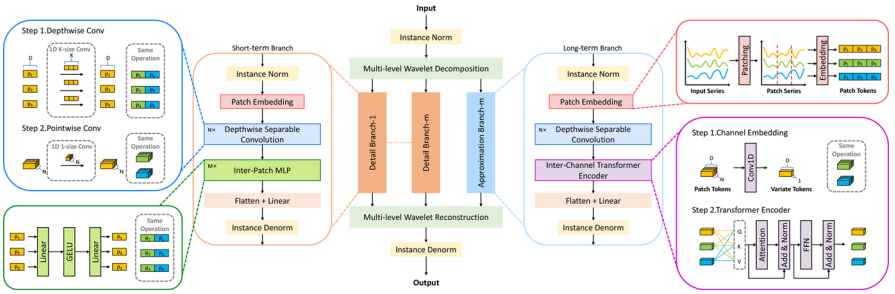


Fig. 3 Overall architecture of WDBNet

Following the decomposition, the modeling pipelines of the two branches operate independently. While both branches apply identical functional steps–including instance normalization, patching and embedding, and depthwise separable convolution–they diverge fundamentally in their feature extraction strategies. Specifically, the long-term branch utilizes a channel-dependent inter-channel encoder to process approximation coefficients, whereas the short-term branch employs a channel-independent patch-wise MLP to process detail coefficients. Finally, the predicted coefficients are reconstructed into the complete cloud workload sequence via the inverse wavelet transform.

To explicitly clarify the novelty of WDBNet, we distinguish our newly introduced designs from reused existing techniques: DWT for sequence decomposition [10], RevIN to mitigate distribution shift [37], Patching for local semantic extraction [20], and DSC for efficient feature extraction [38]. Crucially, we fundamentally differentiate our architecture from existing decomposition-based models and multi-scale networks. While advanced decomposition models like FEDformer [21] leverage frequency-domain attention, they typically apply uniform multivariate channel-mixing strategies across different components, overlooking the distinct inter-channel correlation behaviors at different scales. Similarly, standard multi-scale architectures like MSCNet [36] capture multi-scale patterns by processing the highly mixed original signal simultaneously via parallel convolutions, lacking explicit physical separation of frequencies. In contrast, to address the inherent frequency heterogeneity of cloud workloads, WDBNet adopts a differential dual-branch paradigm. It strictly isolates frequencies via DWT first, and then explicitly decouples channel strategies: applying a channel-dependent strategy for correlated low-frequency trends, and a channel-independent strategy for isolated high-frequency bursts. This combination of physical separation and customized modeling effectively prevents feature confounding, enabling highly efficient extraction of complex temporal patterns.

The detailed descriptions of each module are provided in Sections 3.3 and 3.4.

3.3 Data preprocessing

3.3.1 Reversible instance normalization

The cloud workload sequences typically exhibit high non-stationarity, which can easily lead to data distribution shift issues. To address this challenge, Kim et al. [37] proposed Reversible Instance Normalization (RevIN) to reduce the impact of distribution shift on model performance. The overall process of RevIN can be represented by the following Eq. (1):

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

where i denotes the i-th dimension of the cloud workload input. Our model applies RevIN before decomposition, after reconstruction, and within the branches to eliminate scale differences, mitigate data drift, and improve modeling performance. The specific locations of RevIN are shown in Fig. 3.

3.3.2 Wavelet decomposition

To effectively model the non-stationary and bursty nature of cloud workloads, choosing an appropriate signal decomposition method is critical. While methods like the Fourier Transform (FFT) and Empirical Mode Decomposition (EMD) are widely used, we specifically adopt the Discrete Wavelet Transform (DWT) due to its unique theoretical advantages. First, unlike FFT which loses time-domain localization, or Short-Time Fourier Transform (STFT) which relies on a fixed window size, DWT provides multi-resolution analysis. It offers precise time localization for high-frequency bursty requests and accurate frequency localization for low-frequency long-term trends. Second, compared to heuristic methods like EMD or VMD that often suffer from mode mixing, high computational overhead, and reliance on predefined hyperparameters, DWT is mathematically rigorous and highly efficient with $O(N)$ complexity. This strictly deterministic and lightweight nature makes DWT a highly robust module for our proposed WDBNet.

Leveraging these theoretical advantages, we apply multi-level DWT independently to each of the C server signals along the temporal dimension. This independent decomposition ensures that every individual server is separated into its own low-frequency approximation and high-frequency detail components, strictly preserving its unique

physical characteristics. Specifically, the signal of each server is recursively decomposed. At each level, the approximation coefficients $X_{A_{m-1}}$ are further split into the current level's approximation coefficients X_{A_m} and detail coefficients X_{D_m}, resulting in one set of approximation coefficients and m sets of detail coefficients. The decomposition process is shown in Eq. (2).

$$X_{A_m}, X_{D_m}, ..., X_{D_1} = \text{Decomposition}(X_r, \psi, m), \tag{2}$$

where X_r denotes the instance-normalized workload sequence, ψ represents the type of wavelet basis function, and m indicates the number of wavelet decomposition levels. The approximation coefficients X_{A_i} and detail coefficients $X_{D_i} \in \mathbb{R}^{C \times L_i}$ correspond to the i-th decomposition level, where L_i denotes the sequence length in the time dimension. It should be noted that the lengths of the coefficient sequences differ across decomposition levels in multi-level DWT.

3.3.3 Channel strategy

The choice of channel strategy has a significant impact on model performance. Most existing cloud workload prediction methods [31][35] employ the channel-dependent strategy for modeling, emphasizing inter-channel relationship learning. However, recent studies [20][39] demonstrate that the channel-independent strategy can achieve inter-channel isolation, effectively mitigating the effects of distribution shift. Based on these findings, we further explore an organic integration of both strategies to enable more comprehensive modeling.

We find that WDBNet inherently possesses advantages for combining both channel strategies. Specifically, the approximation coefficients characterize long-term trends where inter-channel correlations exist, making them suitable for channel-dependent modeling, while the detail coefficients reflect short-term fluctuations with weaker cross-channel dependencies, being more appropriate for channel-independent modeling. To validate this modeling hypothesis, we further introduce normalized mutual information (NMI) [40] to analyze dependency levels between channels across different coefficients. The NMI is defined as:

$$NMI(X;Y) = \frac{2I(X;Y)}{H(X) + H(Y)}, \tag{3}$$

where X and Y are discrete random variables. $I(X;Y)$ denotes the mutual information measuring the dependency between X and Y. $H(X)$ and $H(Y)$ represent the entropy quantifying the information content of X and Y, respectively. As shown in Fig. 4, the approximation coefficients from both datasets exhibit consistently higher NMI scores across channels compared to the detail coefficients, indicating stronger long-term trend dependencies and weaker short-term fluctuation correlations. Consequently, we adopt the channel-dependent strategy for approximation coefficients while employing the channel-independent strategy for detail coefficients. Algorithm 1 and Algorithm 2 presents the distinct branches for modeling these two types of coefficients.

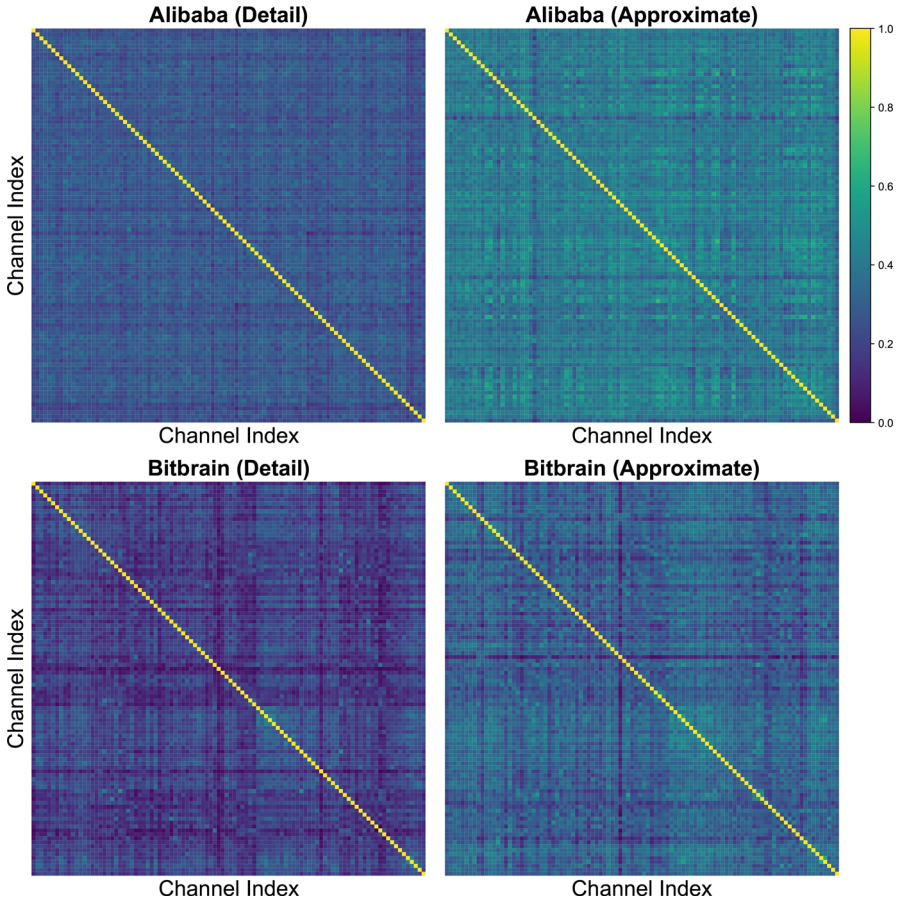


Fig. 4 NMI of approximation coefficients and detail coefficients for different variables in Alibaba and Bitbrain datasets

3.4 Prediction model

3.4.1 Patching and embedding

To effectively capture local sequence features while reducing model parameters, we adopt the same Patching and Embedding technique as PatchTST [20]. For clarity, we demonstrate this process using the approximation coefficients $X_{A_m} \in \mathbb{R}^{C \times l}$ obtained from wavelet decomposition. Let P denote the patch length and S represent the patch stride. X_{A_m} is divided into a series of overlapping patches X_p, where N indicates the number of patches calculated as $N = \left\lfloor \frac{(l-P)}{S} \right\rfloor + 2$. Subsequently, each patch undergoes an embedding operation through a linear projection layer to the model dimension D, producing X_{emb} for higher-order feature extraction. The complete Patching and

Algorithm 1 Long-term branch

Input: approximation coefficients $X_{A_m} \in \mathbb{R}^{C \times l}$
Output: predicted approximation coefficients $Y_{A_m} \in \mathbb{R}^{C \times t}$
Initialize number of Depthwise Separable Convolution Blocks I, patch length P, patch stride S, number of patches $N = \left\lfloor \frac{(l-P)}{S} \right\rfloor + 2$.
Utilize instance normalization to approximation coefficients
$X_r = \frac{X_{A_m} - \mathbb{E}[X_{A_m}]}{\sqrt{\text{Var}[X_{A_m}] + \epsilon}}$.
Perform patching and embedding on the approximation coefficients
$X_{emb} = \text{Linear}(\text{Patching}(X_r, N))$.
for each Depthwise Separable Convolution Block $i \in \{1, ..., I\}$ **do**
  for each patch of coefficients $n \in \{1, ..., N\}$ **do**
    Depthwise Convolution extracts local features within each patch
    $X_{dc} = \text{Conv1D}(X_{emb}, kernel = S, step = 1)$.
    Pointwise Convolution integrates information across patches
    $X_{dsc} = \text{Conv1D}(X_{dc}, kernel = step = 1)$.
  end for
end for
Channel Embedding and dimension reduction
$X_{ce} = \text{Conv1D}(X_{dsc}, kernel = step = 1)$.
$X_k, X_v = \text{Conv1D}(X_{ce})$.
Calculate the inter-channel attention
$X_{attn} = \text{Attention}(X_{ce} W_h^Q, X_k W_h^K, X_v W_h^V)$.
Calculate the Long-term Branch's output
$Y_l = \text{Linear}(\text{Flatten}(X_{attn}))$.
$Y_{A_m} = \sqrt{\text{Var}[X_{A_m}] + \epsilon} \cdot Y_l + \mathbb{E}[X_{A_m}]$.
return Y_{A_m}

Embedding operation can be formally expressed as:

$$\begin{aligned} X_p &= \text{Patching}(X_{A_m}, N) \in \mathbb{R}^{C \times N \times P}, \\ X_{emb} &= \text{Linear}(X_p) \in \mathbb{R}^{C \times N \times D}. \end{aligned} \tag{4}$$

3.4.2 Depthwise separable convolution

To enhance the model's local modeling capacity while maintaining parameter efficiency, we utilize depthwise separable convolution (DSC) modules [38] into both branches. Compared to standard convolution, DSC preserves local modeling capabilities while significantly reducing computational complexity. In our design, DSC operates along the patch dimension. Specifically, the DSC implementation consists of two submodules:

Depthwise Convolution: At this stage, convolution is performed along the feature dimension of each patch to model local temporal dependencies within patches. The process sequentially consists of 1D convolution, GELU activation, and batch normalization, as formally expressed by the following equation:

$$X_{dc}^{(i)} = \text{BN}(\text{GELU}(\text{Conv1D}(X_{emb}^{(i)}, kernel = S, step = 1))), \tag{5}$$

where $X_{emb}^{(i)}$ denotes the patch representation after embedding for the i-th channel, with a kernel size of S and a default convolution stride of 1.

Pointwise Convolution: This stage utilizes 1×1 convolution to fuse features across patches, modeling short-range dependencies between patches to enhance local contextual awareness. The process is shown as follows:

$$X_{dsc}^{(i)} = \text{BN}(\text{GELU}(\text{Conv1D}(X_{dc}^{(i)}, kernel = step = 1))). \tag{6}$$

3.4.3 Inter-patch MLP

To enhance the short-term branch's capability in modeling high-frequency short-term patterns, we introduce a lightweight and efficient Inter-Patch MLP module. This module is specifically designed to capture global dependencies among different patches, thereby compensating for DSC's limitations in modeling non-local information. The computational process is formulated as follows:

$$\begin{aligned} H_{d_1}^{(j)} &= \text{Linear}(X_{ip}^{(j-1)}) \in \mathbb{R}^{C \times D \times N \cdot factor}, \\ H_{d_2}^{(j)} &= \text{GELU}(H_{d_1}^{(j)}), \\ X_{ip}^{(j)} &= \text{Linear}(H_{d_2}^{(j)}) \in \mathbb{R}^{C \times D \times N}, \end{aligned} \tag{7}$$

where $X_{ip}^{(j-1)} \in \mathbb{R}^{C \times D \times N}$ denotes the output of the previous Inter-Patch MLP block.At the first layer, $X_{ip}^{(0)}$ is initialized as X_{dsc}.

3.4.4 Inter-channel transformer encoder

To enhance the long-term branch's capability in modeling persistent trends, we introduce an Inter-Channel Transformer Encoder module to capture global dependencies across channels. This module is integrated after the DSC layer. Specifically, the Inter-Channel Transformer Encoder includes three key components:

Channel Embedding: We perform channel embedding to compress and aggregate patch features. The complete process is formulated as follows:

$$X_{ce} = \text{Conv1D}(X_{dsc}, kernel = step = 1) \in \mathbb{R}^{C \times D}. \tag{8}$$

Channel Dimensionality Reduction: To mitigate overfitting risks in attention mechanisms caused by redundant information from high-dimensional channels while reducing computational complexity, we apply 1D convolution along the channel dimension to compress the key (K) and value (V) representations prior to their generation, reducing the number of channels to $r (r \leq C)$. For instance, on the Alibaba dataset, we employ 1D convolution with a kernel size of 4 and padding of 2 for this compression. This channel dimensionality reduction decreases the computational complexity of attention from $O(C^2 \cdot D)$ to $O(C \cdot r \cdot D)$, as formally expressed below:

$$X_k, X_v = \text{Conv1D}(X_{ce}) \in \mathbb{R}^{r \times D}. \tag{9}$$

Algorithm 2 Short-term branch

Input: detail coefficients $X_{D_m} \in \mathbb{R}^{C \times l}$
Output: predicted detail coefficients $Y_{D_m} \in \mathbb{R}^{C \times t}$
Initialize number of Depthwise Separable Convolution Blocks I, number of Inter-Patch MLP Block J, patch length P, patch stride S, number of patches $N = \left\lfloor \frac{(l-P)}{S} \right\rfloor + 2$.
Utilize instance normalization to approximation coefficients
$X_r = \frac{X_{D_m} - \mathbb{E}[X_{D_m}]}{\sqrt{\text{Var}[X_{D_m}] + \epsilon}}$.
Perform patching and embedding on the approximation coefficients
$X_{emb} = \text{Linear}(\text{Patching}(X_r, N))$.
for each Depthwise Separable Convolution Block $i \in \{1, ..., I\}$ **do**
  for each patch of coefficients $n \in \{1, ..., N\}$ **do**
    Depthwise Convolution extracts local features within each patch
    $X_{dc} = \text{Conv1D}(X_{emb}, kernel = S, step = 1)$.
    Pointwise Convolution integrates information across patches
    $X_{dsc} = \text{Conv1D}(X_{dc}, kernel = step = 1)$.
  end for
end for
for each Inter-Patch MLP Block $j \in \{1, ..., J\}$ **do**
  Further captures global dependencies among patches
  $X_{ip} = \text{Linear}(\text{GELU}(\text{Linear}(X_{dsc}))$
end for
Calculate the Short-term Branch's output
$Y_l = \text{Linear}(\text{Flatten}(X_{ip}))$.
$Y_{D_m} = \sqrt{\text{Var}[X_{D_m}] + \epsilon} \cdot Y_l + \mathbb{E}[X_{D_m}]$.
return Y_{D_m}

Multi-Head Attention: Given the superior global modeling capability of the multi-head attention mechanism, we employ it to model the approximation coefficients. Before performing the attention computation, we first derive the Q_h, K_h, V_h matrices as follows: $Q_h = X_{ce} W_h^Q$, $K_h = X_k W_h^K$, $V_h = X_v W_h^V$, where $h = 1...H$ denotes the number of attention heads. The attention calculation is formally expressed as:

$$\text{Attention}(Q_h, K_h, V_h) = \text{SoftMax}(\frac{Q_h K_h^T}{\sqrt{d}}) V_h. \tag{10}$$

Following the attention computation, the outputs undergo subsequent processing through a feed-forward network (FFN) and Batch Normalization (BatchNorm) operations. The FFN further enhances nonlinear modeling capabilities, thereby improving the model's capacity to fit complex patterns, while BatchNorm stabilizes feature distributions and mitigates training instability caused by distribution shift. The final output is obtained as $X_{attn} \in \mathbb{R}^{C \times D}$.

3.4.5 Wavelet reconstruction

After predicting both the approximation and detail coefficients, the final cloud workload sequence is reconstructed using the following formulation:

$$Y_r = \text{Reconstruction}(Y_{A_m}, Y_{D_m}, ..., Y_{D_1}), \tag{11}$$

Algorithm 3 Wavelet dual-branch network for workload prediction

Input: historical cloud workload $X \in \mathbb{R}^{L \times C}$
Output: future cloud workload $Y \in \mathbb{R}^{T \times C}$
Initialize look-back window size L, horizon window size T, workload channels C, Wavelet decomposition level M.
Utilize instance normalization to historical cloud workload, then transpose dimensions L and C
$X_r = \frac{X - \mathbb{E}[X]}{\sqrt{\text{Var}[X] + \epsilon}} \in \mathbb{R}^{C \times L}$.
for each channels of cloud workload $c \in \{1, ..., C\}$ **do**
  Decomposed into detail and approximation coefficients through an M-level wavelet transform
  for each decomposition level $m \in \{1, .., M\}$ **do**
    Call Algorithm 2 to obtain the Short-term Branch's output
    Y_{D_m} = Short-term Branch(X_{D_m}).
    if m=M **then**
      Call Algorithm 1 to obtain the Long-term Branch's output
      Y_{A_m} = Long-term Branch(X_{A_m}).
    end if
  end for
  Reconstruction the detail and approximation coefficients to obtain the prediction
  Y_r = Reconstruction($Y_{A_m}, Y_{D_m}, ..., Y_{D_1}$) $\in \mathbb{R}^{C \times T}$
end for
Transpose dimensions T and C, then apply inverse instance normalization to the future cloud workload
$Y = \sqrt{\text{Var}[X] + \epsilon} \cdot Y_r + \mathbb{E}[X] \in \mathbb{R}^{T \times C}$.
return Y

where Y_{A_m} and Y_{D_i} denote the predicted values of approximation coefficients and detail coefficients, respectively. The reconstructed sequence is $Y_r \in \mathbb{R}^{C \times T}$, where C represents the number of channels and T denotes the length of the predicted sequence. The key procedures of WDBNet are summarized in Algorithm 3.

4 Experiments

This section first presents the experimental setup. Subsequently, we compare the performance of various models across four datasets and validate the effectiveness of each component through ablation studies. Finally, model analysis is conducted, followed by auto-scaling experiments to evaluate the practical application of WDBNet. All experiments in this section are implemented in PyTorch and executed on three NVIDIA A10 GPUs equipped with 24 GB of memory.

4.1 Experimental setup

Datasets: We evaluate and analyze the performance of our proposed model using four real-world cloud workload datasets: Alibaba, Azure, Bitbrain, and Google.

- Alibaba Cluster Traces [41]: The Alibaba Cluster Trace dataset (Cluster-trace-v2018), collected in 2018, comprises operational data from approximately 4,000 machines over 8 days, including both long running applications (LRA) and batch

workloads with a 10-second sampling interval. The dataset is publicly accessible on GitHub.[1]

- Azure Traces [42]: The Azure Traces dataset (Azure Public Dataset V1) contains a representative subset of first-party Azure VM workloads from a specific geographical region. The dataset includes approximately 2 million virtual machines (VMs) monitored over 30 days, recording key metrics such as VM CPU utilization and VM information tables with a 5-minute sampling interval. The complete dataset is available on GitHub.[2]
- Bitbrains Fast Storage Traces [43]: The Bitbrains Fast Storage Traces dataset contains monitoring data from 1,250 virtual machines (VMs) connected to a fast Storage Area Network (SAN) over 30 days, capturing essential metrics including CPU utilization, memory usage, and other system parameters with a 5-minute sampling interval. This dataset is publicly available on Kaggle.[3]
- Google Cluster Traces [44]: The Google Cluster Trace dataset (clusterdata-2011–2) contains 29 days of operational data from Borg cells in a cluster of approximately 12,500 machines, collected since May 2011 with a 5-minute sampling interval. The dataset is publicly accessible on GitHub.[4]

CPU utilization serves as the core performance metric that directly reflects workload intensity and computational resource pressure. Consequently, we adopt CPU utilization as the primary indicator for assessing workload levels. First, we randomly selected 100 machines from each dataset mentioned above and extracted their CPU utilization to construct the evaluation dataset. Subsequently, the datasets are divided into three subsets: 70% for training, 10% for validation, and 20% for testing.

Baselines: To validate the effectiveness of our approach, we compare it with several state-of-the-art models for both cloud workload prediction and general long-term time series prediction.

- MSCNet [36]: A multi-scale convolutional network specifically designed for long-term cloud workload prediction. Through its multi-scale blocks, MSCNet effectively captures long-term dependencies and periodic patterns in cloud workloads, thereby enhancing both prediction accuracy and generalization capability.
- iTransformer [22]: A Transformer-based model specifically designed for long-term time series prediction. By shifting the attention mechanism from the temporal dimension to the variate dimension, it more naturally captures multivariate correlations, thereby enhancing generalization capability and efficiency for long-term predictions.
- TSMixer [18]: An all-MLP architecture for time series prediction. It captures temporal dependencies and cross-variate interactions through alternating time-mixing and feature-mixing operations. The model eliminates complex modules, relying exclusively on MLPs to deliver efficient and highly generalizable predictions.
- TimesNet [14]: A time series prediction method based on multi-periodicity and 2D convolutions. It transforms one-dimensional sequences into multi-period

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

[2] https://github.com/Azure/AzurePublicDataset

[3] https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains

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

two-dimensional tensors, enabling unified modeling of both intra-period and inter-period variations, and has demonstrated strong performance across various forecasting tasks.

- FEDformer [21]: A frequency-enhanced Transformer model for long-term time series prediction. It combines seasonal-trend decomposition with frequency-domain attention to efficiently capture long-range dependencies and improve forecasting performance.
- DWT-BiGRU-attention [33]: A cloud workload forecasting method combining frequency-domain analysis with recurrent neural networks. It decomposes non-stationary workload sequences into sub-sequences of varying frequencies via discrete wavelet transformation (DWT), and utilizes an attention-mechanism-enhanced BiGRU to extract temporal dependencies, achieving excellent performance in short-term host load prediction tasks.
- EvoGWP [35]: A graph neural network-based(GNN) approach for long-term cloud workload prediction. It automatically extracts shapelets to identify resource usage patterns, and leverages evolutionary graphs and spatiotemporal GNN to accurately capture workload dynamics and enhance prediction accuracy.
- SG-CBA [32]: A workload prediction method integrating Savitzky-Golay filters, convolutional neural networks, and attention-based BiLSTM. SG-CBA achieves superior prediction accuracy compared to multiple baseline models through smoothing preprocessing and deep feature extraction.
- esDNN [31]: A deep neural network-based method for multivariate cloud workload prediction, which integrates a sliding window mechanism with an enhanced GRU to capture high-dimensional temporal features. This approach improves training efficiency and model stability while maintaining prediction accuracy.

Metrics: To accurately evaluate the performance of the models, we adopt three commonly used metrics: mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). The formulas for these metrics are as follows:

$$\begin{aligned} \text{MSE} &= \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \\ \text{MAE} &= \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|, \\ \text{RMSE} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}, \end{aligned} \tag{12}$$

where Y_i and $\hat{Y}_i$ denote the true and predicted values, respectively. n represents the total number of evaluated data points. Specifically, for a test set with N samples, C channels, and a prediction length of T, the total number of points is $n = N \times C \times T$. These three metrics measure the error between predicted and true values, each with distinct characteristics. MSE computes the average squared error and is sensitive to outliers. MAE calculates the average absolute error and is more robust to outliers.

Table 2 Hyperparameter search space and optimal configurations

Hyperparameter	Search Space	Optimal Value
Hidden dimension	{64, 128, 256, 512}	128
Initial learning rate	$\{1 \times 10^{-3}, 5 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-4}\}$	2×10^{-4}
Batch size	{16, 32, 64}	32
Dropout rate	{0.1, 0.2, 0.3}	0.1
Number of Inter-Patch MLP Blocks	{1, 2, 3}	1
Number of DSC Blocks	{1, 2, 3}	1

RMSE is the square root of MSE, combining the characteristics of both and sharing the same unit as the original data. Lower values of these metrics indicate better model performance.

Configuration: To evaluate the performance of various models for long-term cloud workload prediction, we set the historical window size $L = 96$ and prediction windows to {24, 48, 72, 96}. During training, we employ the L2 loss function and the Adam optimizer with an initial learning rate of 2×10^{-4} and a learning rate decay of 0.5. The batch size is set to 32, dropout rate to 0.1, and hidden layer dimension to 128. The model undergoes 100 training epochs with early stopping (patience = 3 epochs) to prevent overfitting. For hyperparameter tuning, we adopt a heuristic search algorithm to determine optimal configurations. Table 2 presents the hyperparameters considered in the search process, their corresponding search spaces, and the optimal configurations obtained for our model.

4.2 Experimental results

To evaluate the performance of different models in cloud workload prediction, we conduct experiments on four datasets: Alibaba, Azure, Bitbrain, and Google, using MSE, MAE, and RMSE as evaluation metrics. The complete prediction results are presented in Table 3. To ensure the reliability of our evaluations, all reported numbers are the average of three independent runs. Furthermore, to demonstrate model stability while maintaining visual clarity in the table, we specifically denote the exact error range for our proposed method. Our experimental results demonstrate that general long-term time series prediction models (e.g., TSMixer, TimesNet) overall outperform specialized cloud workload prediction models (e.g., EvoGWP, SG-CBA, esDNN). This is attributed to the former's incorporation of advanced techniques like RevIN and their superior capability in modeling multi-scale dependencies and periodicity, whereas most specialized models exhibit insufficient robustness against distribution shifts in cloud workloads. Among them, MSCNet currently stands as the best-performing specialized model for cloud workload prediction, with its multi-scale block achieving leading performance on the Alibaba and Google datasets, while the general model iTransformer shows stronger advantages on Azure and Bitbrain. Notably, our proposed WDBNet surpasses both MSCNet and iTransformer across all four datasets, demonstrating significant improvements over other baseline models. Specifically, compared

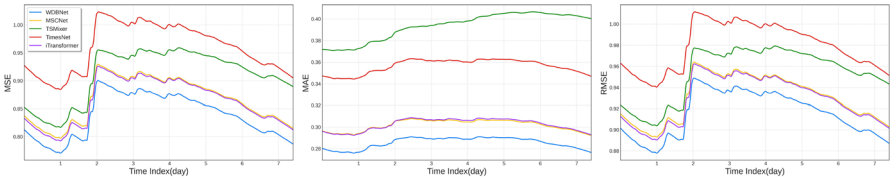
Table 3 Comparison of different models on four datasets across various horizons using mse, mae, and rmse metrics. The best results are highlighted in bold, and the second-best results are underlined

Models		WDBNet			MSCNet			iTransformer			TSMixer			TimesNet		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	**0.3369** ± 0.0002	**0.4163** ± 0.0001	**0.5804** ± 0.0002	0.3423	0.4196	0.5851	0.3502	0.4214	0.5917	0.3537	0.4365	0.5947	0.4104	0.4699	0.6406
	48	**0.3970** ± 0.0002	**0.4591** ± 0.0002	**0.6301** ± 0.0001	0.4018	0.4620	0.6339	0.4093	0.4652	0.6398	0.4351	0.4886	0.6596	0.4518	0.4976	0.6721
	72	**0.4317** ± 0.0002	**0.4827** ± 0.0003	**0.6570** ± 0.0001	0.4370	0.4851	0.6611	0.4449	0.4890	0.6670	0.4723	0.5099	0.6873	0.4786	0.5116	0.6918
	96	**0.4585** ± 0.0001	**0.4988** ± 0.0002	**0.6771** ± 0.0001	0.4637	0.5016	0.6810	0.4720	0.5060	0.6870	0.5162	0.5333	0.7185	0.5037	0.5266	0.7097
Azure	24	**4.0739** ± 0.0005	**0.4199** ± 0.0010	**2.0184** ± 0.0010	4.2619	0.4545	2.0644	4.1162	0.4285	2.0288	5.1268	0.7123	2.2642	5.2466	0.5486	2.2905
	48	**4.5718** ± 0.0060	**0.4509** ± 0.0040	**2.1381** ± 0.0040	4.6954	0.4721	2.1669	4.6434	0.4638	2.1548	5.0316	0.6654	2.2431	5.4534	0.5521	2.3352
	72	**4.8977** ± 0.0020	**0.4746** ± 0.0030	**2.2130** ± 0.0010	5.0454	0.5028	2.2462	5.0287	0.5071	2.2424	5.3636	0.6976	2.3159	5.7368	0.5759	2.3951
	96	**5.0454** ± 0.0060	**0.4867** ± 0.0040	**2.2461** ± 0.0050	5.1475	0.5088	2.2688	5.2522	0.5188	2.2917	5.5256	0.7078	2.3506	5.6806	0.5716	2.3834
Bitbrain	24	**0.7870** ± 0.0050	**0.2766** ± 0.0020	**0.8871** ± 0.0025	0.8149	0.2921	0.9027	0.8125	0.2928	0.9014	0.8897	0.4002	0.9432	0.9053	0.3469	0.9514
	48	**0.8721** ± 0.0050	**0.3197** ± 0.0050	**0.9338** ± 0.0040	0.9023	0.3324	0.9499	0.9006	0.3258	0.9490	1.0053	0.4555	1.0026	0.9852	0.3805	0.9925
	72	**0.9328** ± 0.0040	**0.3488** ± 0.0020	**0.9658** ± 0.0015	0.9605	0.3601	0.9800	0.9569	0.3571	0.9782	1.0782	0.4887	1.0384	1.0516	0.4038	1.0254
	96	**0.9729** ± 0.0050	**0.3688** ± 0.0020	**0.9863** ± 0.0025	1.0053	0.3812	1.0026	1.0006	0.3763	1.0003	1.1434	0.5152	1.0693	1.0830	0.4171	1.0406
Google	24	**1.0448** ± 0.0050	**0.7338** ± 0.0040	**1.0222** ± 0.0030	1.0710	0.7439	1.0349	1.0648	0.7426	1.0318	1.2854	0.8717	1.1337	1.2971	0.8300	1.1389
	48	**1.2308** ± 0.0070	**0.8018** ± 0.0050	**1.1094** ± 0.0050	1.2582	0.8119	1.1217	1.2614	0.8157	1.1231	1.6234	0.9888	1.2741	1.4569	0.8796	1.2070
	72	**1.3664** ± 0.0020	**0.8516** ± 0.0010	**1.1689** ± 0.0010	1.3930	0.8616	1.1802	1.4045	0.8682	1.1851	1.9155	1.0826	1.3840	1.5694	0.9146	1.2527
	96	**1.4775** ± 0.0040	**0.8925** ± 0.0020	**1.2155** ± 0.0020	1.4967	0.9008	1.2234	1.5306	0.9137	1.2371	2.1210	1.1423	1.4563	1.6213	0.9414	1.2733

Table 3 continued

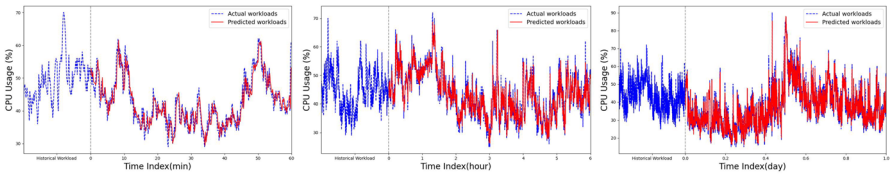
Models		FEDformer			DWT-BiGRU-attention			EvoGWP			SG-CBA			esDNN		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	0.4303	0.4870	0.6559	0.4515	0.4964	0.6719	0.5080	0.5351	0.7127	0.7061	0.6242	0.8403	0.7222	0.6312	0.8498
	48	0.4737	0.5157	0.6881	0.4913	0.5200	0.7009	0.5397	0.5497	0.7346	0.7218	0.6304	0.8496	0.7793	0.6582	0.8828
	72	0.5018	0.5302	0.7083	0.5185	0.5346	0.7201	0.5584	0.5577	0.7473	0.7283	0.6334	0.8534	0.7806	0.6622	0.8835
	96	0.5281	0.5458	0.7266	0.5381	0.5460	07335	0.5723	0.5626	0.7565	0.7220	0.6286	0.8497	0.8990	0.7174	0.9481
Azure	24	5.3038	0.6083	2.3030	5.3471	0.5577	2.3123	5.7946	0.7230	2.4072	5.9494	0.6734	2.4391	6.2023	0.7682	2.4904
	48	5.5494	0.6108	2.3557	5.7166	0.5811	2.3909	5.8412	0.7276	2.4168	5.9614	0.6856	2.4416	6.2156	0.7632	2.4931
	72	5.6986	0.6169	2.3871	5.9180	0.6098	2.4327	5.9107	0.7302	2.4311	6.0401	0.6918	2.4576	6.2957	0.7754	2.5091
	96	5.7732	0.6204	2.4027	5.8375	0.5957	2.4161	5.9973	0.7331	2.4489	6.1811	0.7182	2.4861	6.3737	0.7849	2.5246
Bitbrain	24	0.9971	0.4071	0.9985	0.9395	0.3698	0.9692	1.2644	0.6018	1.1244	1.3395	0.6139	1.1573	1.3396	0.6084	1.1574
	48	1.0177	0.4238	1.0088	1.0071	0.3903	1.0035	1.3212	0.6225	1.1494	1.3978	0.6459	1.1822	1.3752	0.6386	1.1727
	72	1.0697	0.4418	1.0343	1.0640	0.4170	1.0315	1.3628	0.6365	1.1673	1.4152	0.6698	1.1896	1.4014	0.6542	1.1838
	96	1.1272	0.4712	1.0617	1.1085	0.4372	1.0528	1.3840	0.6520	1.1764	1.4323	0.6736	1.1968	1.4199	0.6651	1.1915
Google	24	1.3431	0.8515	1.1589	1.3311	0.8430	1.1537	2.5049	1.2496	1.5827	2.9030	1.3406	1.7038	3.0062	1.3684	1.7338
	48	1.4567	0.8891	1.2069	1.5349	0.9114	1.2389	2.6248	1.2787	1.6201	2.9805	1.3540	1.7264	3.3119	1.4446	1.8198
	72	1.5528	0.9233	1.2461	1.6156	0.9378	1.2710	2.7467	1.3086	1.6573	3.1499	1.4000	1.7748	3.2874	1.4407	1.8131
	96	1.6307	0.9476	1.2770	1.6636	0.9525	1.2898	2.8380	1.3297	1.6846	3.3164	1.4458	1.8211	3.3766	1.4623	1.8375

The prediction window size $T \in \{24, 48, 72, 96\}$ is set for all datasets



(a) MSE (b) MAE (c) RMSE

Fig. 5 Detdetailed MSE, MAE and RMSE results of different models with various time index on the Bitbrain dataset



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

Fig. 6 Prediction performance display of WDBNet on the Alibaba dataset with various levels of time index

Table 4 Detailed wilcoxon signed-rank test results for mse over 10 random seeds (WDBNet vs. MSCNet)

Dataset	Horizon (T)	W-Statistic	p-value	Significance ($\alpha = 0.05$)
Azure	24	0.00	< 0.001	Yes
	48	2.00	0.0029	Yes
	72	1.00	0.0020	Yes
	96	0.00	< 0.001	Yes
Google	24	0.00	< 0.001	Yes
	48	0.00	< 0.001	Yes
	72	0.00	< 0.001	Yes
	96	0.00	< 0.001	Yes

One-tailed Wilcoxon signed-rank test. A p-value < 0.05 indicates that the predictive performance of WDBNet is significantly better than that of MSCNet

to MSCNet, WDBNet reduces MSE by 1.3%, 3.1%, 3.3%, and 2.0% on Alibaba, Azure, Bitbrain, and Google datasets, respectively. Furthermore, WDBNet achieves optimal performance across all evaluation metrics under various prediction windows, validating the strong capability of its wavelet-based dual-branch differential modeling strategy in capturing both long-term and short-term dependencies.

To further verify the statistical significance of WDBNet's improvements over MSCNet, we obtained paired MSE observations by evaluating both models across 10 independent random seeds on the Azure and Google datasets. We then applied a one-tailed Wilcoxon signed-rank test to these paired results for each dataset-horizon setting. As shown in Table 4, all obtained p-values are below the significance level of 0.05 across the four prediction horizons on both datasets. This indicates that WDBNet

achieves statistically significant MSE reductions compared with MSCNet, and the observed improvements are unlikely to be merely caused by random seed selection.

To evaluate the performance of different models over the continuous testing period, we plot the curves of MSE, MAE, and RMSE on the Bitbrain dataset against the chronological time index, as shown in Fig. 5. It is important to note that the time index represents the progression of days within the test set, not the prediction horizon (T). Results show that WDBNet consistently outperforms others across all metrics and time indices, maintaining its advantage as the evaluation timeline extends, which demonstrates strong stability and robustness on long-term test data.

Figure 6 illustrates WDBNet's prediction performance on minute-level, hour-level, and day-level workloads from the Alibaba dataset. The blue dashed lines represent ground truth, while the red solid lines denote predictions. WDBNet accurately captures both long-term trends and short-term fluctuations. To further highlight model differences, Fig. 7 presents hour-level prediction curves on the Google dataset for the top four models. Compared to others, WDBNet achieves the best fit on sudden changes, demonstrating its short-term branch's superior responsiveness to high-frequency variations. Furthermore, to provide a comprehensive understanding of the model's behavior under extreme scenarios, a targeted analysis of the failure cases in Fig. 7(a) reveals both its strengths and practical limitations. While the dual-branch design effectively avoids over-smoothing by strictly preserving the exact timing and directional shifts of rapid bursts, it exhibits amplitude damping during unprecedented extreme events. For instance, during the extreme spike at a Time Index of around 1.2 h and the abrupt plunge at around 1.9 h, WDBNet successfully anticipates the occurrence and trend of these structural changes but underestimates their absolute magnitudes. This conservative prediction behavior illustrates a common trade-off in deep learning, where the model inherently constrains extreme outliers to maintain global structural stability.

To validate the transferability and generalization capability of WDBNet, we train all models on the Alibaba dataset and evaluate them on three distinct datasets: Azure, Bitbrain, and Google. As presented in Table 5, WDBNet achieves either optimal or sub-optimal performance across all metrics, demonstrating significant improvements over competing models. These results confirm that its wavelet decomposition and dual-branch differential modeling strategy exhibit strong cross-scenario adaptability.

4.3 Ablation studies

To evaluate the effectiveness of individual modules in WDBNet and validate the impact of different channel strategies on performance, we design two categories of ablation studies: module ablation experiments and channel strategy comparison experiments.

For the first category, to assess the specific contributions of individual components, we conduct a series of tests by progressively removing key modules. Specifically, we construct several model variants by omitting the multi-scale wavelet decomposition, patch embedding, depthwise separable convolution, inter-channel Transformer encoder, and inter-patch MLP. To isolate the effect of frequency decoupling and ensure a parameter-fair comparison, the "W/O DWT" variant feeds the raw time series directly into both branches. The outputs are then fused via feature concatenation

Table 5 The experimental results of transferring models trained on the alibaba dataset to the azure, bitbrain and google datasets. The best results are highlighted in bold, and the second-best results are underlined

Models		WDBNet			MSCNet			iTransformer			TSMixer			TimesNet		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Azure	24	**4.2805**	**0.4736**	**2.0689**	4.4574	0.5210	2.1112	4.5368	0.5175	2.1299	4.4114	0.6061	2.1003	5.1245	0.5714	2.2637
	48	**4.7283**	**0.4903**	**2.1744**	4.9502	0.5507	2.2249	5.0235	0.5405	2.2413	5.0659	0.7261	2.2507	5.4632	0.5822	2.3373
	72	**5.0487**	**0.5089**	**2.2469**	5.2322	0.5747	2.2874	5.2728	0.5555	2.2962	5.3862	0.7811	2.3208	5.7824	0.5982	2.4046
	96	**5.2450**	**0.5176**	**2.2902**	5.4086	0.5922	2.3256	5.4078	0.5635	2.3254	5.6657	0.8480	2.3802	5.8440	0.6078	2.4174
Bitbrain	24	0.8520	**0.3199**	0.9230	**0.8519**	0.3313	**0.9229**	0.8612	0.3329	0.9280	0.8883	0.4196	0.9425	1.0032	0.3954	1.0016
	48	**0.9413**	**0.3554**	**0.9702**	0.9424	0.3703	0.9707	0.9519	0.3707	0.9756	1.0450	0.5179	1.0222	1.0517	0.4161	1.0255
	72	1.0114	**0.3843**	1.0057	**1.0069**	0.3996	**1.0034**	1.0120	0.3969	1.0060	1.1159	0.5596	1.0563	1.1295	0.4414	1.0628
	96	**1.0539**	**0.4028**	**1.0266**	1.0562	0.4221	1.0277	1.0574	0.4160	1.0283	1.1979	0.6014	1.0944	1.2110	0.4672	1.1004
Google	24	**1.0701**	**0.7454**	**1.0344**	1.0721	0.7482	1.0354	1.0863	0.7559	1.0422	1.1502	0.8076	1.0724	1.3352	0.8446	1.1555
	48	1.2816	**0.8213**	1.1321	**1.2712**	0.8218	**1.1274**	1.2865	0.8274	1.1343	1.5072	0.9418	1.2276	1.4715	0.8940	1.2130
	72	1.4155	**0.8685**	1.1897	**1.4086**	0.8716	**1.1868**	1.4217	0.8762	1.1923	1.6863	1.0018	1.2986	1.6150	0.9403	1.2708
	96	1.5208	**0.9072**	1.2332	**1.5132**	0.9107	**1.2301**	1.5232	0.9136	1.2341	1.8908	1.0662	1.3750	1.7454	0.9842	1.3211

Table 5 continued

Models		FEDformer			DWT-BiGRU-attention			EvoGWP			SG-CBA			esDNN		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Azure	24	5.1804	0.6336	2.2761	5.5292	0.5925	2.3514	5.5318	0.8009	2.3519	6.3980	1.0255	2.5294	6.4570	0.9634	2.5410
	48	5.5594	0.6441	2.3578	5.9431	0.6147	2.4378	5.7257	0.8425	2.3928	6.4674	1.0642	2.5431	6.4415	0.9761	2.5380
	72	5.7439	0.6408	2.3966	6.1780	0.6387	2.4855	5.8838	0.8683	2.4256	6.5646	1.0909	2.5621	6.5050	0.9832	2.5504
	96	5.9393	0.6597	2.4370	6.1368	0.6369	2.4772	6.0069	0.8840	2.4509	6.5677	1.0655	2.5627	6.4509	0.9347	2.5398
Bitbrain	24	1.1049	0.4640	1.0512	1.0778	0.4170	1.0382	1.2156	0.6169	1.1025	1.5348	0.7693	1.2388	1.6482	0.8228	1.2838
	48	1.0864	0.4635	1.0423	1.1191	0.4393	1.0578	1.2940	0.6570	1.1375	1.5826	0.7819	1.2580	1.6135	0.8044	1.2702
	72	1.1489	0.4829	1.0720	1.2004	0.4667	1.0956	1.3474	0.6840	1.1607	1.6488	0.8131	1.2840	1.6412	0.8158	1.2811
	96	1.2604	0.5278	1.1227	1.2268	0.4768	1.1076	1.3881	0.7045	1.1782	1.6845	0.8319	1.2979	1.5712	0.7686	1.2535
Google	24	1.3826	0.8665	1.1758	1.4470	0.8819	1.2029	1.9361	1.0975	1.3914	2.5566	1.2534	1.5989	2.7842	1.3114	1.6686
	48	1.4713	0.9037	1.2129	1.5676	0.9205	1.2520	2.1325	1.1538	1.4603	2.5928	1.2573	1.6102	3.0218	1.3743	1.7383
	72	1.5979	0.9492	1.2641	1.7003	0.9687	1.3039	2.2329	1.1829	1.4943	2.6892	1.2932	1.6399	2.9853	1.3688	1.7278
	96	1.7555	0.9907	1.3249	1.7806	0.9946	1.3344	2.2863	1.1991	1.5120	2.6611	1.2861	1.6313	2.9917	1.3734	1.7296

Table 6 Ablation study of WDBNet'S different modules on four cloud workload datasets. The best results are highlighted in bold, and the second-best results are underlined

Models		WDBNet			W/O_DWT			W/O_Patch			W/O_Trans			W/O_DSC			W/O_MLP		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	**0.3369**	0.4163	**0.5804**	0.3379	**0.4158**	0.5813	0.3381	0.4164	0.5815	0.3396	0.4165	0.5828	0.3384	0.4171	0.5817	0.3376	0.4165	0.5810
	48	**0.3970**	**0.4591**	**0.6301**	0.3973	0.4594	0.6303	0.3978	0.4598	0.6307	0.3993	0.4601	0.6319	0.3978	0.4596	0.6307	0.3980	0.4601	0.6308
	72	**0.4317**	0.4827	**0.6570**	0.4330	0.4830	0.6580	0.4323	0.4828	0.6575	0.4348	0.4830	0.6594	0.4321	**0.4823**	0.6573	0.4329	0.4830	0.6580
	96	**0.4585**	0.4988	**0.6771**	0.4603	0.4997	0.6784	0.4590	**0.4986**	0.6775	0.4618	0.5002	0.6796	0.4588	0.4992	0.6773	0.4595	0.5000	0.6779
Azure	24	**4.0739**	**0.4199**	**2.0184**	4.1415	0.4376	2.0350	4.0850	0.4297	2.0211	4.1255	0.4341	2.0311	4.0992	0.4302	2.0246	4.1242	0.4291	2.0308
	48	**4.5718**	0.4509	**2.1381**	4.6048	0.4616	2.1459	4.6408	0.4671	2.1542	4.6506	0.4686	2.1565	4.6561	0.4656	2.1578	4.5849	**0.4495**	2.1412
	72	**4.8977**	**0.4746**	**2.2130**	4.9100	0.4783	2.2158	4.9177	0.4857	2.2175	4.9130	0.4839	2.2165	4.9248	0.4853	2.2191	4.9094	0.4808	2.2157
	96	**5.0454**	**0.4867**	**2.2461**	5.0885	0.4935	2.2557	5.1322	0.5112	2.2654	5.1005	0.4982	2.2584	5.0739	0.4981	2.2525	5.0680	0.4946	2.2512
Bitbrain	24	**0.7870**	0.2766	**0.8871**	0.7984	0.2842	0.8935	0.7970	0.2858	0.8927	0.7940	0.2800	0.8911	0.7918	**0.2759**	0.8898	0.7896	0.2814	0.8886
	48	**0.8721**	0.3197	**0.9338**	0.8883	0.3244	0.9425	0.8773	0.3275	0.9366	0.9003	0.3241	0.9488	0.8799	**0.3170**	0.9380	0.8742	0.3203	0.9350
	72	0.9328	**0.3488**	0.9658	0.9656	0.3564	0.9826	0.9468	0.3573	0.9730	0.9467	0.3539	0.9730	0.9461	0.3537	0.9727	**0.9318**	0.3492	**0.9653**
	96	**0.9729**	**0.3688**	**0.9863**	1.0011	0.3789	1.0005	0.9770	0.3742	0.9884	0.9873	0.3724	0.9936	0.9942	0.3744	0.9971	0.9789	0.3708	0.9894
Google	24	1.0448	0.7338	1.0222	1.0500	0.7366	1.0247	**1.0443**	0.7327	**1.0219**	1.0508	0.7366	1.0251	1.0537	0.7375	1.0265	1.0472	0.7323	1.0233
	48	**1.2308**	**0.8018**	**1.1094**	1.2600	0.8128	1.1225	1.2414	0.8077	1.1141	1.2449	0.8077	1.1157	1.2370	0.8073	1.1122	1.2354	0.8050	1.1114
	72	**1.3664**	**0.8516**	**1.1689**	1.3868	0.8598	1.1776	1.3751	0.8549	1.1726	1.3845	0.8602	1.1766	1.3794	0.8590	1.1745	1.3884	0.8598	1.1783
	96	**1.4775**	**0.8925**	**1.2155**	1.4952	0.9024	1.2228	1.4819	0.8957	1.2173	1.4872	0.8988	1.2195	1.4856	0.8991	1.2188	1.4794	0.8943	1.2163

Table 7 Ablation study of WDBNet'S channel strategy on four cloud workload datasets. The best results are highlighted in bold, and the second-best results are underlined

Case	Long-term Branch CI or CD	Short-term Branch CI or CD	Alibaba			Azure			Bitbrain			Google		
			MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
1	CI	CI	0.4088	0.4649	0.6384	4.6974	0.4712	2.1656	0.9070	0.3326	0.9516	1.2918	0.8258	1.1342
2	CD	CI	**0.4060**	0.4642	**0.6361**	**4.6472**	**0.4580**	**2.1539**	**0.8912**	**0.3284**	**0.9432**	**1.2798**	**0.8199**	**1.1290**
3	CI	CD	0.4072	**0.4638**	0.6370	4.6799	0.4668	2.1616	0.9046	0.3303	0.9503	1.2919	0.8259	1.1342
4	CD	CD	0.4068	0.4647	0.6367	4.6779	0.4652	2.1610	0.9145	0.3324	0.9555	1.2806	0.8222	1.1293

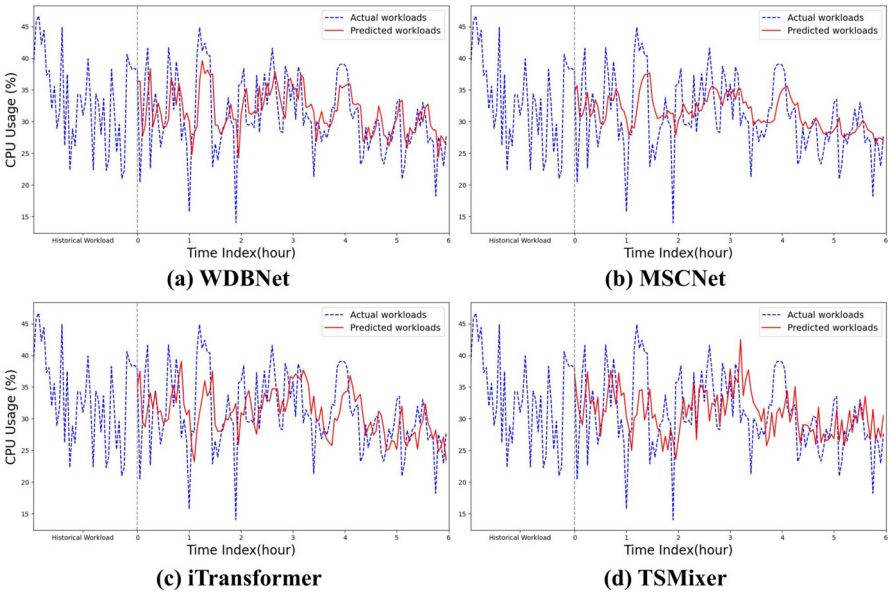
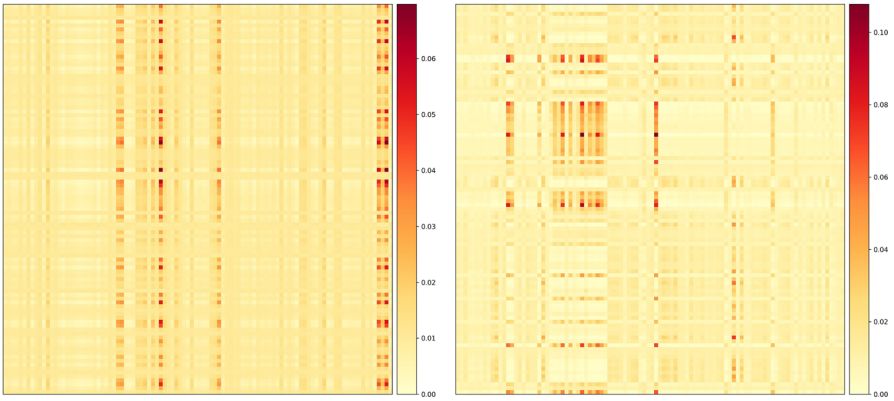


Fig. 7 Prediction performance display of different models on the Google dataset with hour-level time index

and a linear projection layer, functioning as a standard dual-stream ensemble without explicit frequency separation. These variants are evaluated across four datasets. Table 6 summarizes the impact of each module on overall model performance. The experimental results demonstrate that multi-scale wavelet decomposition and inter-channel attention are most critical for performance gains, highlighting the importance of multi-scale modeling and capturing inter-variable dependencies. Patch embedding not only improves prediction accuracy but also significantly enhances training efficiency. Meanwhile, depthwise separable convolution and inter-patch MLP contribute to modeling local temporal patterns while keeping the model lightweight.

Furthermore, we compare four channel modeling strategies: (1) both branches adopt channel-independent (CI) modeling, (2) the long-term branch uses channel-dependent (CD) modeling while the short-term branch uses CI, (3) the long-term branch uses CI while the short-term branch uses CD, and (4) both branches adopt CD modeling. For clarity, the results are presented as average metrics across different prediction horizons, as shown in Table 7. Experimental results show that applying CI in the long-term branch yields significantly worse performance than CD, indicating the existence of long-term inter-channel correlations within the cluster. Modeling inter-variable dependencies helps capture trend patterns more effectively. In contrast, using CI in the short-term branch outperforms CD, possibly because short-term bursts and fluctuations tend to be independently distributed across channels, and CD may introduce redundant interference. In summary, CD is more suitable for modeling long-term inter-channel dependencies in the long-term branch, while CI is preferable in the short-term branch for capturing localized intra-channel patterns.



(a) Alibaba **(b) Bitbrain**

Fig. 8 Channel attention heatmaps of the long-term branch on the bitbrain and alibaba datasets

4.4 Model analysis

4.4.1 Analysis of the inter-channel transformer encoder

To investigate the actual utilization of inter-channel information by the Long-term Branch, we visualize the channel attention heatmaps generated by its Inter-Channel Transformer Encoder on the Bitbrain and Alibaba datasets (as shown in Fig. 8). The Bitbrain heatmap exhibits a strongly clustered inter-channel attention structure, demonstrating that the model actively captures and reinforces the strong inter-channel correlations present in the data. In contrast, the Alibaba heatmap displays a dispersed yet selective weight distribution, indicating that the attention mechanism can still identify key channels even when the underlying correlations are relatively weak. Overall, this analysis convincingly demonstrates the data-adaptiveness of the Long-term Branch's Inter-Channel Transformer Encoder. It can dynamically adjust its focusing pattern based on the strength of inter-channel correlations, thereby effectively utilizing the inter-channel statistical structure for feature aggregation.

4.4.2 Impact of different decomposition levels

The number of decomposition levels plays a crucial role in the performance of WDBNet, and selecting an appropriate level helps maximize the model's modeling capability. As shown in Fig. 9, we present the average MSE performance under different values of m across four datasets, and additionally compare the results with the variant that does not use wavelet decomposition ($m = 0$).

The experiments demonstrate that the optimal performance is achieved when $m = 1$, indicating that a single-level wavelet decomposition effectively separates high- and low-frequency information while avoiding excessive dilution of information. The inferior performance of deeper decompositions ($m = 2 \sim 4$) compared to $m = 1$ is likely because the primary trends and fluctuations are typically concentrated in the

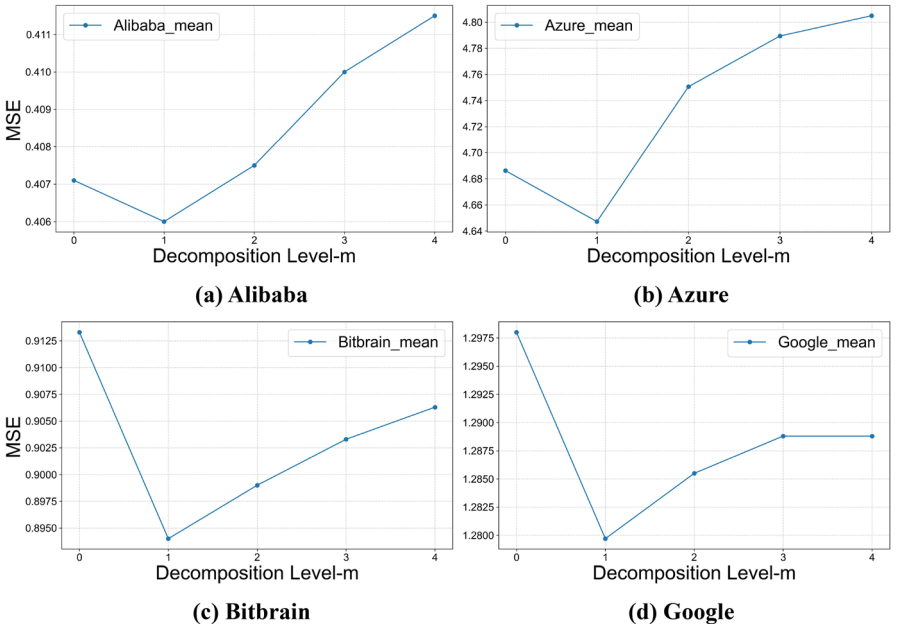


Fig. 9 The prediction accuracy varies with the decomposition level-m

low-frequency and first-level high-frequency components. On the Bitbrain and Google datasets, even though excessive decomposition slightly degrades performance, it still outperforms the case without wavelet decomposition ($m = 0$). This phenomenon validates the effectiveness of wavelet decomposition in extracting multi-frequency features.

4.4.3 Impact of wavelet basis functions

To evaluate the robustness of WDBNet to the choice of the wavelet basis function ψ, we conducted a comprehensive sensitivity analysis on the representative Google dataset. Keeping the decomposition level fixed at $m = 1$, we evaluated six widely used wavelet bases characterized by distinct time-frequency localization properties and vanishing moments: Daubechies (db3, db4), Symlets (sym7, sym8), and Coiflets (coif2, coif3).

As shown in Table 8, we tested these configurations across all four prediction windows ($T \in \{24, 48, 72, 96\}$). The results demonstrate that the predictive performance (MSE, MAE, and RMSE) remains highly stable across all selected bases, with only marginal fluctuations. Crucially, regardless of the specific mathematical properties of the chosen wavelet, WDBNet consistently maintains high prediction accuracy. This empirical evidence validates that the performance gains of our model stem inherently from the proposed heterogeneous dual-branch frequency-decoupling architecture, rather than being implicitly tuned or overfitted to a specific wavelet configuration.

Table 8 Sensitivity analysis of different wavelet basis functions on the google dataset

Prediction Window Size		$T = 24$			$T = 48$			$T = 72$			$T = 96$		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Wavelet Basis	db3	1.0501	0.7357	1.0247	1.2384	0.8068	1.1128	1.3720	0.8546	1.1713	1.4808	0.8975	1.2168
	db4	1.0499	0.7374	1.0246	1.2321	0.8032	1.1100	1.3749	0.8546	1.1726	1.4771	0.8946	1.2153
	sym7	1.0492	0.7410	1.0243	1.2386	0.8065	1.1129	1.3682	0.8528	1.1697	1.4732	0.8940	1.2137
	sym8	1.0517	0.7374	1.0255	1.2464	0.8087	1.1164	1.3713	0.8548	1.1710	1.4720	0.8924	1.2132
	coif2	1.0560	0.7383	1.0276	1.2388	0.8064	1.1130	1.3729	0.8556	1.1717	1.4813	0.8983	1.2171
	coif3	1.0506	0.7333	1.0250	1.2376	0.8049	1.1124	1.3715	0.8538	1.1711	1.4802	0.8938	1.2166

Table 9 The prediction accuracy varies with parameter N. The best results are highlighted in bold

Number of DSC		N=1			N=2			N=3		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	0.3369	0.4163	0.5804	0.3368	**0.4162**	0.5803	**0.3365**	0.4163	**0.5801**
	48	**0.3970**	**0.4591**	**0.6301**	0.3972	0.4593	0.6302	0.3974	0.4604	0.6304
	72	**0.4317**	0.4827	**0.6570**	0.4321	**0.4824**	0.6573	0.4324	0.4825	0.6575
	96	**0.4585**	**0.4988**	**0.6771**	0.4589	0.4992	0.6774	0.4592	0.4992	0.6776
Azure	24	4.0739	0.4199	2.0184	**4.0635**	**0.4188**	**2.0158**	4.0904	0.4248	2.0224
	48	**4.5718**	0.4509	**2.1381**	4.5736	0.4513	2.1386	4.5810	**0.4486**	2.1403
	72	**4.8977**	**0.4746**	**2.2130**	4.9127	0.4864	2.2164	4.9149	0.4826	2.2169
	96	**5.0454**	**0.4867**	**2.2461**	5.0769	0.4970	2.2532	5.0765	0.4905	2.2531
Bitbrain	24	**0.7831**	**0.2732**	**0.8849**	0.7961	0.2777	0.8922	0.7910	0.2783	0.8894
	48	**0.8774**	0.3222	**0.9366**	0.8805	0.3177	0.9383	0.8776	**0.3176**	0.9368
	72	**0.9348**	**0.3484**	**0.9668**	0.9349	0.3492	0.9669	0.9362	0.3505	0.9676
	96	0.9808	0.3712	0.9903	**0.9762**	**0.3671**	**0.9880**	0.9768	0.3682	0.9883
Google	24	**1.0406**	**0.7319**	**1.0201**	1.0537	0.7376	1.0265	1.0525	0.7372	1.0259
	48	1.2373	0.8062	1.1123	1.2390	0.8058	1.1131	**1.2369**	**0.8053**	**1.1121**
	72	**1.3676**	**0.8521**	**1.1694**	1.3737	0.8563	1.1720	1.3762	0.8567	1.1731
	96	**1.4735**	**0.8932**	**1.2139**	1.4904	0.8979	1.2208	1.4846	0.8953	1.2184

4.4.4 Impact of the number of depthwise separable convolution(DSC) blocks

The DSC Block is employed to efficiently extract local patterns in the short-term branch, which is particularly crucial for modeling abrupt behavioral changes. As shown in Table 9, the model achieves optimal performance when the number of blocks $N = 1$, while further increasing the block count leads to a slight performance degradation. This indicates that an appropriate number of DSC blocks facilitates the modeling of local features, but excessive stacking not only increases model complexity but may also induce overfitting to local patterns, introducing redundant features and noise, thereby impairing overall performance.

4.4.5 Impact of the number of inter-patch MLP blocks

The Inter-Patch MLP enhances the modeling capability of dependencies among patches in the short-term branch, facilitating the capture of nonlinear abrupt patterns. As presented in Table 10, the model achieves optimal performance with layer depth $M = 1$ on the Alibaba and Azure datasets, whereas it performs best at $M = 3$ for Bitbrain and Google. This demonstrates that the Inter-Patch MLP effectively improves representational capacity, yet its optimal depth should be adaptively determined based on data characteristics to balance generalization ability and modeling complexity.

4.4.6 Efficiency analysis

In addition to prediction accuracy, computational efficiency is a critical metric for real-world deployment. As shown in Table 11, we conduct a comprehensive comparison of parameters, multiply-accumulate operations (MACs), and average time overheads. Compared to high-capacity models like MSCNet and iTransformer, WDBNet achieves significantly lower parameter counts and MACs. While it exhibits a slightly more complex architecture than lightweight baselines such as TSMixer and EvoGWP, this marginal increase in computational overhead is well justified by the substantial improvements in prediction accuracy, demonstrating an excellent balance between efficiency and performance.

Beyond theoretical metrics, actual deployment feasibility hinges on inference latency across different operational scenarios. In a centralized cloud environment prioritizing high-concurrency throughput (GPU, Batch Size = 32), the total inference time for a 1-level decomposition is 15.50 ms, with DWT/IDWT accounting for only 9.81%. Even at a 3-level cascade, where boundary padding effects increase the wavelet time proportion to 18.91%, the amortized per-sample cost remains highly efficient. Furthermore, to validate its applicability for decentralized edge deployment, we simulated a resource-constrained environment strictly limited to a single CPU thread with a Batch Size of 1. Given that WDBNet empirically achieves its optimal predictive performance under a 1-level decomposition, we specifically adopted this configuration for the edge simulation. Under this rigorous setting, the pure single-sample inference latency is remarkably low at 12.73 ms (with DWT/IDWT taking 1.66 ms), decisively proving that WDBNet comfortably satisfies the strict real-time processing threshold for edge devices.

Table 10 The prediction accuracy varies with parameter M. The best results are highlighted in bold

Number of MLP		M=1			M=2			M=3		
Metric		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Alibaba	24	0.3369	0.4163	**0.5804**	**0.3368**	**0.4162**	0.5805	0.3370	0.4165	0.5808
	48	**0.3970**	**0.4591**	**0.6301**	0.3971	0.4595	0.6303	0.3971	0.4593	0.6304
	72	0.4317	0.4827	**0.6570**	0.4318	0.4825	0.6574	**0.4316**	**0.4822**	0.6572
	96	**0.4585**	**0.4988**	**0.6771**	0.4587	0.4992	0.6775	0.4588	0.4991	0.6776
Azure	24	**4.0739**	**0.4199**	**2.0184**	4.0755	0.4204	2.0188	4.0886	0.4219	2.0220
	48	**4.5718**	**0.4509**	**2.1381**	4.6165	0.4650	2.1486	4.5906	0.4534	2.1425
	72	4.8977	0.4746	2.2130	**4.8844**	**0.4707**	**2.2100**	4.8929	0.4731	2.212
	96	**5.0454**	**0.4867**	**2.2461**	5.0827	0.4980	2.2544	5.0549	0.4895	2.2483
Bitbrain	24	**0.7831**	**0.2732**	**0.8849**	0.7885	0.2745	0.8879	0.787	0.2766	0.8871
	48	0.8774	0.3222	0.9366	0.8767	0.321	0.9363	**0.8721**	**0.3197**	**0.9338**
	72	0.9348	0.3484	0.9668	0.9334	**0.3482**	0.9661	**0.9328**	0.3488	**0.9658**
	96	0.9808	0.3712	0.9903	0.9759	0.3703	0.9879	**0.9729**	**0.3688**	**0.9863**
Google	24	**1.0406**	**0.7319**	**1.0201**	1.0426	0.7337	1.021	1.0448	0.7338	1.0222
	48	1.2373	0.8062	1.1123	1.2348	0.8041	1.1112	**1.2308**	**0.8018**	**1.1094**
	72	1.3676	0.8521	1.1694	1.3665	0.8522	1.1689	**1.3664**	**0.8516**	**1.1689**
	96	**1.4735**	0.8932	**1.2139**	1.4748	0.8936	1.2144	1.4775	**0.8925**	1.2155

Table 11 Number of trainable parameters, macs, training time per epoch, and inference time of different models on the azure dataset

Models	Parameters	MACs	Train Time	Inference Time
WDBNet	488.35 K	69.62 M	6.20 s	15.50 ms
MSCNet	830.98 K	216.92 M	46.60 s	16.22 ms
iTransformer	3.22 M	321.64 M	3.98 s	12.14 ms
TSMixer	103.58 K	10.25 M	2.97 s	10.68 ms
TimesNet	18.81 M	11.25 G	25.13 s	22.19 ms
FEDformer	17.3 M	791.24 M	59.94 s	56.78 ms
DWT-BiGRU-attention	1.48 M	74.52 M	21.57 s	23.55 ms
EvoGWP	124.9 K	9.61 M	3.47 s	11.83 ms
SG-CBA	280.63 K	32.55 M	11.47 s	21.15 ms
esDNN	170.72 K	14.96 M	9.08 s	22.11 ms

4.5 Applying WDBNet for machines auto-scaling with simulations

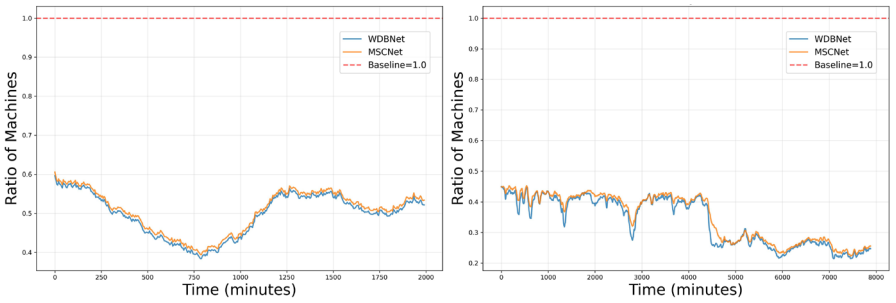
The auto-scaling technique can dynamically adjust the number of active physical machines based on the real-time system status. Specifically, it reduces the number of active machines to save resources when resource utilization is low and increases the machine count to ensure service quality when the system load is high. A well-designed auto-scaling strategy can significantly improve resource utilization and overall system performance.

To further evaluate the practical value of WDBNet in real-world cloud data center environments, this paper integrates it into the physical machine auto-scaling scenarios of Alibaba and Google data centers for simulation experiments. The experiments are based on the real machine specifications provided by the datasets, with a scheduling cycle set to 5 min. Our goal is to use accurate CPU utilization predictions to reduce the number of active machines required, thereby enhancing resource utilization.

Inspired by the auto-scaling process in esDNN[31], we use the predicted CPU utilization as the input to the auto-scaling module to compute the required number of active machines at each time step. At the same time, we employ a baseline method for comparison, where the average number of active machines in previous time slices is used as the predicted value for the current time. The calculation formula is as follows:

$$M_{base}(t) = \frac{1}{m} \sum_{i=1}^{m} M_{base}(t-i), \tag{13}$$

where $M_{base}(t)$ represents the number of active machines at time t, and m is the number of historical time slices used. In our experimental setup, we set $m = 5$. Subsequently, we calculate the ratio of active machines between the WDBNet auto-scaling scheme and the baseline scheme. If this ratio is less than 1.0, it indicates that WDBNet is able to reduce the number of active machines compared to the baseline method.



(a) Optimization ratio of Alibaba machines **(b) Optimization ratio of Google machines**

Fig. 10 Optimization ratio comparison in minutes with different traces

Figure 10 presents the auto-scaling experiment results for the Alibaba and Google datasets. On the Alibaba dataset, WDBNet and MSCNet reduce the number of active machines by approximately 40% to 60%, while on the Google dataset, this reduction further increases to 55% to 80%. Additionally, WDBNet consistently achieves a lower ratio than MSCNet, indicating its superior ability to save active machines and improve resource utilization efficiency across both datasets.

5 Conclusion

Accurate long-term cloud workload prediction is essential for proactive resource management, ensuring QoS, and minimizing resource waste. In this work, we propose WDBNet, an effective and well-balanced approach that achieves strong predictive performance while maintaining high computational efficiency. By leveraging frequency-aware modeling and a tailored dual-branch design, WDBNet captures long-term inter-channel dependencies and short-term localized dynamics more effectively than existing methods. Extensive evaluations on four real-world datasets demonstrate that WDBNet consistently outperforms state-of-the-art approaches, striking a superior balance between accuracy and efficiency. Additionally, we applied WDBNet to auto-scaling scenarios, where the results show that it significantly reduces the number of active machines and improves resource utilization efficiency, showcasing strong applicability and real-time performance for resource management in real-world high-performance computing (HPC), distributed, and supercomputing cloud environments.

Beyond cloud workload prediction, WDBNet may also be applicable to other long-term time-series forecasting tasks characterized by non-stationarity and multi-scale temporal patterns, such as energy demand forecasting, network traffic prediction, traffic flow forecasting, and industrial monitoring. Its wavelet-based decomposition can separate long-term trends from short-term fluctuations, while the dual-branch channel strategy can flexibly model both cross-variable dependencies and variable-specific dynamics. Future studies may further examine its effectiveness and adaptability in these broader time-series domains.

In the future, our work will focus on three key directions: adaptive modeling, holistic multi-metric forecasting, and model lightweighting. First, while the current WDBNet employs predefined wavelet basis functions and decomposition levels, future studies could investigate data-driven adaptive wavelet transforms. Second, although this study primarily demonstrates the model's effectiveness on CPU utilization, extending the proposed framework to predict other critical workload metrics–such as memory utilization and I/O throughput–represents a highly valuable direction for comprehensive resource management. Finally, to meet the low-latency prediction requirements of ultra-large-scale clusters, further optimization of the model's inference efficiency will constitute another critical research direction.

Acknowledgements This work was supported by the Guangxi Key Research and Development Project (Grant No. 2024AB02018), the China Postdoctoral Science Foundation Project (2023M741242), the Innovative Development Joint Fund of the Natural Science Foundation of Shandong Province (Grant No. ZR2024LZH012), the Guangdong Provincial Natural Science Foundation Project (Grant No. 2025A1515010113), and the Major Key Project of Pengcheng Laboratory, China (Grant No. PCL2025A08 and PCL2025A11).

Author Contributions Zhenkai Yuan was involved in conceptualization, methodology, and writing original draft preparation. Bo Liu and Weiwei Lin took part in experiment, formal analysis, and writing, review and editing.Shaomin Tang and Keqin Li performed supervision and writing, review and editing.

Data Availability The source of all datasets used in this study is given in the manuscript.

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

References

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53(4):50–58
2. Moreno IS, Garraghan P, Townend P, Xu J (2013) An approach for characterizing workloads in google cloud to derive realistic resource utilization models. IEEE Seventh International Symposium on Service-oriented System Engineering 49–60IEEE
3. Li S, Wang Y, Qiu X, Wang D, Wang L (2013) A workload prediction-based multi-vm provisioning mechanism in cloud computing. In: 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–6. IEEE
4. Bi J, Yuan H, Li S, Zhang K, Zhang J, Zhou M (2024) Arima-based and multiapplication workload prediction with wavelet decomposition and savitzky-golay filter in clouds. IEEE Transact Syst Man Cybernet Syst 54(4):2495–2506
5. Nikravesh AY, Ajila SA, Lung C-H (2015) Towards an autonomic auto-scaling prediction system for cloud resource provisioning. IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE, pp 35–45
6. Yu Q, Yang G, Wang X, Shi Y, Feng Y, Liu A (2025) A review of time series forecasting and spatio-temporal series forecasting in deep learning: Q. yu et al. The Journal of Supercomputing 81(10):1160
7. Zhang Y, Zhong K, Xie X, Huang Y, Han S, Liu G, Chen Z (2025) Vmd-convtsmixer: Spatiotemporal channel mixing model for non-stationary time series forecasting. Expert Syst Appl 271:126535
8. Tian G, Zhang C, Shi Y, Li X (2024) Multiwavenet: A long time series forecasting framework based on multi-scale analysis and multi-channel feature fusion. Expert Syst Appl 251:124088
9. Mallat SG (1989) A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans Pattern Anal Mach Intell 11(7):674–693

10. Joo TW, Kim SB (2015) Time series forecasting based on wavelet filtering. Expert Syst Appl 42(8):3868–3874
11. Chen J, Zhang X, Gao K, Zha D, Han Y, Guo J (2025) Warns: a time series model integrating wavelet convolution and channel attention mechanism. J Supercomput 81(12):1–18
12. Lin S, Lin W, Wu W, Zhao F, Mo R, Zhang H (2023) Segrnn: Segment recurrent neural network for long-term time series forecasting. arXiv preprint arXiv:2308.11200
13. Bergsma S, Zeyl T, Guo L (2023) Sutranets: sub-series autoregressive networks for long-sequence, probabilistic forecasting. Advances in Neural Information Processing Systems, vol 36. pp 30518–30533
14. Wu H, Hu T, Liu Y, Zhou H, Wang J, Long M (2022) Timesnet: Temporal 2d-variation modeling for general time series analysis. arXiv preprint, arXiv:2210.02186
15. Wang H, Peng J, Huang F, Wang J, Chen J, Xiao Y (2023) Micn: Multi-scale local and global context modeling for long-term series forecasting. In: The Eleventh International Conference on Learning Representations
16. Gong Z, Tang Y, Liang J (2023) Patchmixer: A patch-mixing architecture for long-term time series forecasting. arXiv preprint, arXiv:2310.00655
17. Zeng A, Chen M, Zhang L, Xu Q (2023) Are transformers effective for time series forecasting? In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 11121–11128
18. Chen S-A, Li C-L, Yoder N, Arik SO, Pfister T (2023) Tsmixer: An all-mlp architecture for time series forecasting. arXiv preprint, arXiv:2303.06053
19. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115
20. Nie Y, Nguyen NH, Sinthong P, Kalagnanam J (2022) A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint, arXiv:2211.14730
21. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R (2022) Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. Proceedings of the 39th International Conference on Machine Learning. pp 27268–27286 (**PMLR**)
22. Liu Y, Hu T, Zhang H, Wu H, Wang S, Ma L, Long M (2023) itransformer: Inverted transformers are effective for time series forecasting. arXiv preprint, arXiv:2310.06625
23. Liu P, Wu B, Hu Y, Li N, Dai T, Bao J, Xia S-T (2024) Timebridge: Non-stationarity matters for long-term time series forecasting. arXiv preprint, arXiv:2410.04442
24. Naghashi V, Boukadoum M, Diallo AB (2025) A multiscale model for multivariate time series forecasting. Sci Rep 15(1):1565
25. Singh P, Gupta P, Jyoti K (2019) Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. Clust Comput 22(2):619–633
26. Xie Y, Jin M, Zou Z, Xu G, Feng D, Liu W, Long D (2020) Real-time prediction of docker container resource load based on a hybrid model of arima and triple exponential smoothing. IEEE Transactions on Cloud Computing 10(2):1386–1401
27. Kaur G, Bala A, Chana I (2019) An intelligent regressive ensemble approach for predicting resource usage in cloud computing. Journal of Parallel and Distributed Computing 123:1–12
28. Kim IK, Wang W, Qi Y, Humphrey M (2020) Forecasting cloud application workloads with cloudinsight for predictive resource management. IEEE Transactions on Cloud Computing 10(3):1848–1863
29. Barati M, Sharifian S (2015) A hybrid heuristic-based tuned support vector regression model for cloud load prediction. J Supercomput 71:4235–4259
30. Chen Z, Hu J, Min G, Zomaya AY, El-Ghazawi T (2019) Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans Parallel Distrib Syst 31(4):923–934
31. Xu M, Song C, Wu H, Gill SS, Ye K, Xu C (2022) esdnn: deep neural network based multivariate workload prediction in cloud computing environments. ACM Transactions on Internet Technology (TOIT) 22(3):1–24
32. Chen L, Zhang W, Ye H (2022) Accurate workload prediction for edge data centers: Savitzky-golay filter, cnn and bilstm with attention mechanism. Appl Intell 52(11):13027–13042
33. Dogani J, Khunjush F, Seydali M (2023) Host load prediction in cloud computing with discrete wavelet transformation (dwt) and bidirectional gated recurrent unit (bigru) network. Comput Commun 198:157–174
34. Li J, Zhang B (2025) Cloud workload prediction based on wavelet transform noise reduction and a tcn-gru hybrid model. Clust Comput 28(12):767

35. Li J, Yao J, Xiao D, Yang D, Wu W (2024) Evogwp: Predicting long-term changes in cloud workloads using deep graph-evolution learning. IEEE Trans Parallel Distrib Syst 35(3):499–516
36. Zhao F, Lin W, Lin S, Tang S, Li K (2025) Mscnet: Multi-scale network with convolutions for long-term cloud workload prediction. IEEE Transactions on Services Computing
37. Kim T, Kim J, Tae Y, Park C, Choi J-H, Choo J (2021) Reversible instance normalization for accurate time-series forecasting against distribution shift. In: International Conference on Learning Representations
38. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 1251–1258
39. Lin S, Lin W, Zhao F, Chen H (2025) Benchmarking and revisiting time series forecasting methods in cloud workload prediction. Clust Comput 28(1):71
40. Shannon CE (1948) A mathematical theory of communication. The Bell system technical journal 27(3):379–423
41. Guo J, Chang Z, Wang S, Ding H, Feng Y, Mao L, Bao Y (2019) Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. Proceedings of the International Symposium on Quality of Service. pp 1–10
42. Cortez E, Bonde A, Muzio A, Russinovich M, Fontoura M, Bianchini R (2017) Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. Proceedings of the 26th Symposium on Operating Systems Principles. pp 153–167
43. Shen S, Van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, pp 465–474
44. Reiss C, Wilkes J, Hellerstein JL (2011) Google cluster-usage traces: format+ schema. Google Inc., White Paper 1, 1–14