Fremer: Lightweight and Effective Frequency Transformer for Workload Forecasting in Cloud Services

Hengyu Ye[1,*], Jiadong Chen[1,3,*], Fuxin Jiang[2], Xiao He[2],
Tieying Zhang[2], Jianjun Chen[2], Xiaofeng Gao[1,†]
[1] Shanghai Jiao Tong Univeristy; [2] ByteDance Inc.; [3] University of New South Wales;
{cs_22_yhy,chenjiadong998}@sjtu.edu.cn,gao-xf@cs.sjtu.edu.cn
{jiangfuxin,xiao.hx,tieying.zhang,jianjun.chen}@bytedance.com

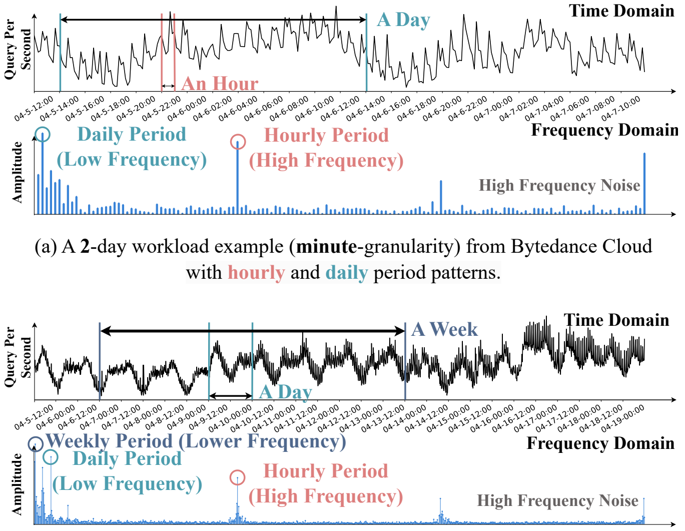
ABSTRACT

Workload forecasting is pivotal in cloud service applications, such as auto-scaling and scheduling, with profound implications for operational efficiency. Although Transformer-based forecasting models have demonstrated remarkable success in general tasks, their computational efficiency often falls short of the stringent requirements in large-scale cloud environments. Given that most workload series exhibit complicated periodic patterns, addressing these challenges in the frequency domain offers substantial advantages. To this end, we propose `Fremer`, an efficient and effective deep forecasting model. `Fremer` fulfills three critical requirements: it demonstrates superior efficiency, outperforming most Transformer-based forecasting models; it achieves exceptional accuracy, surpassing all state-of-the-art (SOTA) models in workload forecasting; and it exhibits robust performance for multi-period series. Furthermore, we collect and open-source four high-quality, open-source workload datasets derived from ByteDance's cloud services, encompassing workload data from thousands of computing instances. Extensive experiments on both our proprietary datasets and public benchmarks demonstrate that `Fremer` consistently outperforms baseline models, achieving average improvements of 5.5% in MSE, 4.7% in MAE, and 8.6% in SMAPE over SOTA models, while simultaneously reducing parameter scale and computational costs. Additionally, in a proactive auto-scaling test based on Kubernetes, `Fremer` improves average latency by 18.78% and reduces resource consumption by 2.35%, underscoring its practical efficacy in real-world applications.

PVLDB Reference Format:
Hengyu Ye, Jiadong Chen, Fuxin Jiang, Xiao He, Tieying Zhang, Jianjun Chen, Xiaofeng Gao. Fremer: Lightweight and Effective Frequency Transformer for Workload Forecasting in Cloud Services. PVLDB, 18(11): 3812 - 3825, 2025.
doi:10.14778/3749646.3749656

PVLDB Artifact Availability:
The source code has been made available at https://github.com/YHYHYHYHYHY/Fremer. The ByteDance Cloud workload datasets have been made available at https://huggingface.co/datasets/ByteDance/CloudTimeSeriesData.

1 INTRODUCTION



(a) A **2**-day workload example (**minute**-granularity) from Bytedance Cloud with hourly and daily period patterns.

(b) A **14**-day workload example (**minute**-granularity) from Bytedance Cloud with hourly, daily, and weekly period patterns.

Figure 1: Workload from cloud services in ByteDance exhibit complicated periodicity, both in time and frequency domain.

In modern cloud service platforms, tens of thousands of applications and millions of microservices are deployed to support diverse workloads. To ensure service quality (QoS) and optimize resource utilization, these platforms rely on monitoring systems that collect and analyze workload patterns. Accurate workload forecasting is critical for enabling predictive active scaling technologies, which are widely adopted to meet dynamic demand [1, 11, 23, 31, 32, 43].

However, achieving both efficiency and effectiveness in workload forecasting poses significant challenges in large-scale cloud environments. For instance, the Platform-as-a-Service (PaaS) of Bytedance Cloud requires the execution of over 100,000 forecasting tasks per hour, demanding efficient models that can operate with minimal computational overhead. Figure 1 illustrates workload series from a real-world cloud computing system, showing minute-granularity patterns over 2 days (Figure 1(a)) and 14 days (Figure 1(b)). The cloud system supports diverse services like web applications, real-time data processing, and machine learning tasks, thus inherently handling highly complex workloads series in the time domain with hourly, daily, and weekly temporal patterns. In

[*] indicates equal contribution. † Xiaofeng Gao is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749656

the frequency domain, different periodic components and high-frequency noise are more easily distinguishable. Moreover, time series data with significant variation and hard-to-quantify similarity in the time domain appears more similar in the frequency domain. Frequency-domain information is often concentrated near a few frequencies and exists in combinations, such as harmonics, as shown in the lower parts of Figure 1(a) and 1(b). These characteristics motivate our use of frequency-domain methods for fine-grained, large-scale workload series forecasting.

Among existing time-series forecasting models, those based on the Transformer architecture have achieved state-of-the-art (SOTA) performance. This is largely attributed to the attention mechanism, which enables effective modeling of long sequences [35, 52, 58, 59]. However, these methods often struggle to capture multi-periodic patterns in complex time series, particularly when fine-grained periodicity (e.g., minute-level) intertwines with broader cycles (e.g., daily or weekly), leading to suboptimal forecasting accuracy, as shown in Figure 2. Meanwhile, the efficiency drawbacks of Transformer limit its deployment in real-world scenarios.

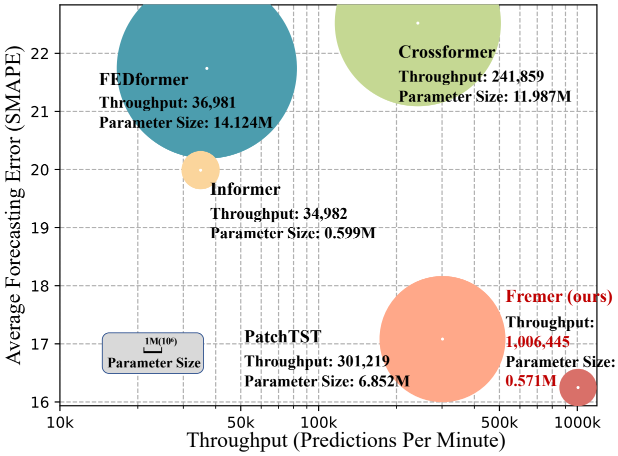


Figure 2: Comparison of throughput and prediction error among different Transformers on IaaS workload dataset (experimental settings are referred to in Section 5.1).

The complexity of attention mechanism is quadratically proportional to the length of the input series, which restricts their deployment in dealing with long sequence predictions. Moreover, the inherent susceptibility of neural networks to overfitting poses significant challenges in achieving robust generalization [57], particularly when handling large-scale and complex time series data. Figure 2 illustrates the throughput and accuracy of different Transformer-based models, as well as their parameter volume. Existing works either have low accuracy or are affected by parameter volume or computational efficiency that affect total throughput.

The efficiency bottleneck of Transformer-based methods lies primarily in the computation of attention. For a sequence of length L, the time and space complexity of the attention mechanism are both $O(L^2)$, resulting in significant inference time and GPU memory usage. Furthermore, beyond the model, the time domain representation of series is also a barrier to efficient and accurate data mining. Research indicates that obtaining a compact representation of sequences in the time domain is relatively difficult [61]. In contrast, sequence data can achieve a compact representation in the frequency domain [60], as noise, trends, and periodic information are effectively decoupled in this domain. Moreover, the frequency domain allows for better capture of global information (such as period shift) [61], as demonstrated in Figure 1.

Transforming time series into the frequency domain can provide a foundation for efficient mining of sequence information, but there are still challenges that need to be addressed. **1) The issue of frequency mis-alignment**. The Discrete Fourier Transform (DFT) is a common method for transforming time series from the time domain to the frequency domain. Since the frequency sampling interval of the DFT is fixed (for a sequence of length L, the sampling interval is $1/L$), the true periodic frequency of the sequence may not be sampled, as shown in Figure 4. However, this issue has not yet received sufficient attention from the community. To the best of our knowledge, we are the first to address this problem. **2) The challenge of extracting key frequency information**. Information in the frequency domain often exists in a combination form (e.g., harmonics). The global information of a sequence often originates from the combination of a few key periodic features, especially when the sequence contains multiple periodic patterns as shown in Figure 1. Recent works [27, 61] model point-to-point relationships between frequency points while neglecting the issue of frequency-domain information appearing as combinations. Therefore, it is necessary to explore the relationships between frequency combinations.

In this paper, we propose Fremer, an efficient and effective forecasting model. Fremer aims at forecasting the spectrum of the complete series (including input part and forecasting part) based on spectrum of input series. Specifically, targeting at the unique characteristics of frequency domain representation, we design **1) Learnable Linear Padding** for addressing the frequency resolution mis-alignment problem; **2) Complex-valued Spectrum Attention** to effectively capture global dependencies from the frequency combinations; **3) Frequency Filters** for handling noise and overfitting.

We also provide and open-source four high-quality workload datasets[1] from distinct cloud service types in ByteDance, containing workload data of thousands of computing instances, spanning from 1 month to 2 months. We carefully preprocess the raw data, and make them useful tools for the community to evaluate and develop new workload forecasting methods. Through extensive experiments on these datasets and three public datasets, our proposed Fremer achieved an average improvement of 5.5% in MSE, 4.7% in MAE and 8.6% in SMAPE compared to current SOTA forecasting models, with scale of parameters and computing costs reduced by times.

Our contributions are summarized as follows:

- We proposed Fremer, a deep forecasting framework for workload forecasting, effectively utilizing the frequency domain representation to achieve the balance among accuracy, efficiency, and generalizability.
- We design Learnable Linear Padding, Frequency Filters, and Complex-valued Spectrum Attention, with which Fremer outperforms all SOTA forecasting models.

[1] https://huggingface.co/datasets/ByteDance/CloudTimeSeriesData

- We open-source four workload datasets from ByteDance's cloud services, containing workload data from thousands of computing instances over 1–2 months, providing robust support for training and evaluating forecasting models.
- Extensive experiments on the Time Series Forecasting Benchmark (TFB)[39] and our datasets demonstrate Fremer 's superiority. It achieves performance improvements across datasets in various domains while significantly reducing parameter scale and computational costs. In Kubernetes HPA proactive auto-scaling tests, Fremer reduces average latency by 18.78% and resource consumption by 2.35%, validating its real-world efficacy.

2 RELATED WORK

Overview of Forecasting Methods. Early time series forecasting predominantly relied on traditional statistical methods, such as ETS [13], ARIMA [5, 6, 17, 21, 25], STL [12], and regression-based methods [2, 4, 40, 54]. However, these methods often have limitations in capturing complex temporal dynamics. In recent years, deep learning has significantly propelled the field of time series forecasting [8, 16, 33, 42]. RNN-based methods [15, 22, 24, 41, 44] and CNN-based methods [3, 28] have shown enhanced abilities in extracting intricate patterns from time series data. Additionally, MLP-based methods [7, 36, 57] offer a straightforward yet effective approach for learning the non-linear temporal dependencies. The frequency domain offers a unique perspective for analyzing time series data by revealing underlying periodicity and frequency components. Several forecasting methods have been developed to exploit these characteristics and capture global dependencies inherent in time series data, such as FECAM [14], FilM [60], FreDo [45], FreTS [56], FilterNet [55] and FITS [53].

Transformer-based Forecasting Methods. Transformers have achieved breakthroughs in many tasks, and attention mechanisms have outperformed traditional time series analysis methods. Based on model architecture and design concepts, we categorize Transformer-based forecasting methods into four groups: long-sequence modeling, sequence decomposition, spatial dependency, and periodicity-aware approaches. Long-sequence modeling methods (Informer [59], PatchTST [35], Reformer [20], Pyraformer [29]) efficiently capture long-range temporal dependencies. Sequence decomposition methods (FEDformer [61], Autoformer [52], ETSformer [50], Basisformer [34]) decompose sequences into trend, periodic, and residual components for higher forecasting accuracy. Spatial dependency methods (Crossformer [58], iTransformer [30], Earthformer [10], Airformer [26]) model variable correlations using spatio-temporal information. Frequency and periodicity-aware methods (Fredformer [38], PDF [9]) extract periodicity information via Discrete Fourier Transform to explore intra-period patterns and inter-period relationships.

3 PRELIMINARY

3.1 Workload Forecasting

Definition of Workload Series In the context of cloud computing, a workload series represents a time-ordered aggregation of workloads generated by jobs or applications operating on cloud infrastructure [1].The Common types of workload in cloud environment, including distinct resource-consumption patterns (such as CPU and memory usage) and user-request metrics (like Queries Per Second-QPS). For a computing instance, the workload series can be mathematically defined as $\mathbf{X} = \{x_0, x_1, \ldots, x_{L-1}\}$, where L denotes the length of a given time period, which is equivalent to the number of time steps utilized as input, and w_i represents the numerical value of the specific workload at time-step i.

Workload Forecasting. Given the historical value of a workload series $\mathbf{X} = \{x_0, x_1, \ldots, x_{L-1}\}$, the objective of workload forecasting is to predict the future workload series $\hat{\mathbf{Y}} = \{x_L, x_{L+1}, \ldots, x_{L+T-1}\}$. In this expression, T represents the length of the forecasting horizon, that is, the number of time steps to be predicted. The target of workload forecasting is to make the prediction $\hat{\mathbf{Y}}$ as accurate as possible, which is equivalent to minimizing the gap between the prediction $\hat{\mathbf{Y}}$ and the corresponding ground truth value $\mathbf{Y}$.

3.2 Time Series in Frequency Domain

The frequency domain of a workload series refers to the representation of the data in terms of its frequency components. The frequency domain analysis provides insights into the underlying periodicity and patterns present in the workload series data that may not be readily apparent in the time domain.

Discrete Fourier Transform The Discrete Fourier Transform (DFT) is a mathematical transformation that converts a time series into its frequency domain representation, revealing the frequency components present in the data [46]. Specifically, for an input series with length L , the DFT spectrum is calculated as follows:

$$\mathbf{F}[k] = \sum_{n=0}^{L-1} \mathbf{X}[n] e^{-2\pi i \frac{kn}{L}}, k = 0, 1, \ldots, L-1. \tag{1}$$

Here, $\mathbf{F}[k]$ represents the frequency-domain representation (the spectrum) at frequency index k, $\mathbf{X}$ represents the time-domain input sequence, L is the number of data points in the sequence, and i is the imaginary unit ($i^2 = -1$).

The Inverse Discrete Fourier Transform (iDFT) is the reverse operation of the DFT. It takes a frequency-domain signal, obtained through the DFT, and reconstructs the original time-domain signal:

$$\mathbf{X}[n] = \frac{1}{L} \sum_{k=0}^{L-1} \mathbf{F}[k] e^{2\pi i \frac{kn}{L}}, n = 0, 1, \ldots, L-1. \tag{2}$$

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT and iDFT. In practice, the Real FFT (rFFT) is typically used for transforming real-valued data, as it exploits Hermitian symmetry to compute only the non-redundant positive frequencies, reducing computational cost and memory usage.

Frequency Filters. Frequency filters selectively enhance or suppress specific frequency components in a signal. A High-Pass Filter (HPF) attenuates low-frequency components, isolating rapid changes or fine details, while a Low-Pass Filter (LPF) attenuates high-frequency components, smoothing noise and capturing broader trends. These filters are indispensable in signal processing, enabling precise control over frequency ranges to meet specific needs.

4 PROPOSED METHOD

In this section, we propose Fremer, which re-designs the classical Transformer into an Encoder-only architecture based on frequency representation. This architecture is specialized for workload forecasting from the perspective of the frequency domain. Fremer incorporates two key designs that effectively address the challenges in frequency-domain forecasting: 1) Learnable Linear Padding (LLP) for frequency resolution alignment; 2) Complex-valued Attention (CSA) for extracting the relationships between frequency combinations and reducing complexity. Moreover, Fremer is designed in a channel-independent manner, and its effectiveness has been demonstrated by [35]. This design also endows Fremer with generalizability, enabling it to be easily applied to workload forecasting for unseen instances.

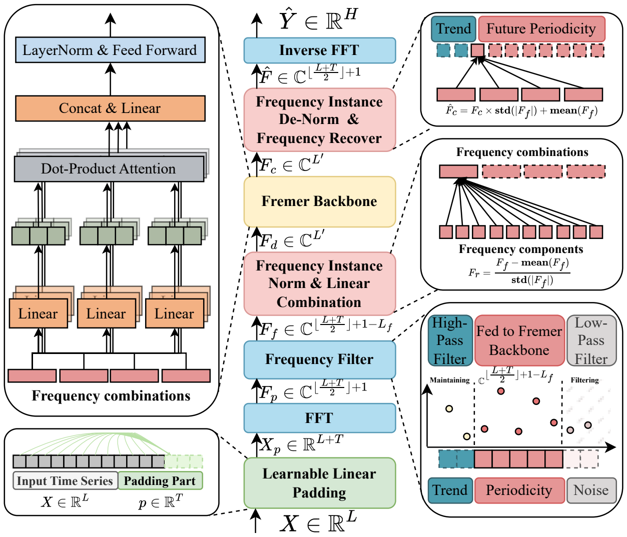

LayerNorm & Feed Forward
Concat & Linear
Dot-Product Attention
Linear
Linear
Linear
Frequency combinations
Input Time Series
Padding Part
$X \in \mathbb{R}^L$
$p \in \mathbb{R}^T$
$\hat{Y} \in \mathbb{R}^H$
Inverse FFT
$\hat{F} \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1}$
Frequency Instance De-Norm & Frequency Recover
$F_c \in \mathbb{C}^{L'}$
Fremer Backbone
$F_d \in \mathbb{C}^{L'}$
Frequency Instance Norm & Linear Combination
$F_f \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1 - L_f}$
Frequency Filter
$F_p \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1}$
FFT
$X_p \in \mathbb{R}^{L+T}$
Learnable Linear Padding
$X \in \mathbb{R}^L$
Trend
Future Periodicity
$\hat{F}_c = F_c \times \mathbf{std}(|F_f|) + \mathbf{mean}(F_f)$
Frequency combinations
Frequency components
$F_r = \frac{F_f - \mathbf{mean}(F_f)}{\mathbf{std}(|F_f|)}$
High-Pass Filter
Fed to Fremer Backbone
Low-Pass Filter
Maintaining
$\mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1 - L_f}$
Filtering
Trend
Periodicity
Noise

Figure 3: Model Architecture of Fremer.

As illustrated in Figure 3, the entire pipeline of Fremer can be described as follows: Given an input workload series $\mathbf{X} \in \mathbb{R}^L$, we use LLP to pad it to $\mathbf{X}_p \in \mathbb{R}^{L+T}$, where T represents the forecasting horizon. Subsequently, $\mathbf{X}_p$ is transformed into the frequency domain using the real-valued Fast Fourier Transform (rFFT), yielding $\mathbf{F} \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1}$. After being processed by Frequency Filters (F-Filter), we apply Frequency Reversible Instance Norm (F-RIN) to $\mathbf{F}$ and then feed it into the Complex-valued Spectrum Attention (CSA) block. Finally, we use the inverse real-valued Fast Fourier Transform (irFFT) to transform it back to the time domain and obtain the forecasting results. In the subsequent part, we will provide a detailed description of the components of Fremer.

4.1 Learnable Linear Padding

Frequency Resolution Alignment. Fremer is designed to forecast the frequency spectrum of the complete series (both the input part and the forecasting part) based on the spectrum of the input series. However, the frequency resolution is determined by the length of the series. As a result, the resolutions of the two spectra are misaligned. As shown in Figure 4, when the resolutions of the spectra are misaligned, it becomes difficult to retrieve the corresponding information at specific frequencies of the complete-series spectrum based on the input spectrum.

The following is a detailed explanation of the Frequency Resolution Alignment problem. According to the formula of DFT:

$$\mathbf{F}[k] = \sum_{n=0}^{L-1} \mathbf{X}[n] e^{-2\pi i \frac{kn}{L}}, k = 0, 1, \dots, L-1,$$

each value in the DFT spectrum represents the inner product between the series ($\mathbf{X}[n]$) and a specific trigonometric basis ($e^{-2\pi i \frac{kn}{L}}$) at a corresponding frequency ($\frac{2\pi k}{L}$). For input series $\mathbf{X}_{inp} \in \mathbb{R}^L$ and complete series $\mathbf{X}_{comp} \in \mathbb{R}^{L+T}$, where L is the look-back window and T is the forecasting horizon, we denote the corresponding DFT spectrum as $\mathbf{F}_{inp} \in \mathbb{C}^L$, $\mathbf{F}_{comp} \in \mathbb{C}^{L+T}$. In the case of $\mathbf{F}_{inp}$, the frequencies are denoted as $f_{inp} \in \{\frac{2\pi k}{L} | k = 0, 1, \dots, L-1\}$, and for $\mathbf{F}_{comp}$, the frequencies are $f_{comp} \in \{\frac{2\pi k}{L+T} | k = 0, 1, \dots, L+T-1\}$.

The misalignment of frequencies refers to a situation where the frequencies in f_{inp} and f_{comp} do not perfectly correspond. In other words, there may be important frequencies present in f_{comp} that are not captured in f_{inp}.

To illustrate this, let's consider the example with an evident period of 24, where $L = 300$ and $T = 60$, as depicted in Figure 4. Our goal is to predict $\mathbf{F}_{comp}[k]$ using $\mathbf{F}_{inp}$, and we focus on the specific period 24, which corresponds to the frequency index 15 in the spectrum of complete series. The ground truth value for $\mathbf{F}_{comp}[15]$ is determined by summing the inner products between $\mathbf{X}_{comp}$ and the trigonometric basis at the corresponding frequency $\frac{\pi}{12}$ ($\frac{2\pi \times 15}{300+60}$). However, when we examine the input spectrum $\mathbf{F}_{inp}$, we find the frequency $\frac{\pi}{12}$ is not present in f_{inp}. The two most adjacent frequencies in f_{inp} are $\frac{2\pi}{25}$ and $\frac{13\pi}{150}$(corresponding to $k = 12, 13$ respectively). This means the dominant period 24 (corresponding to the frequency $\frac{\pi}{12}$) cannot be accurately detected in the spectrum of input series due to the misalignment of frequencies.

Learnable Linear Padding. To tackle the frequency-resolution misalignment problem, we propose the Learnable Linear Padding (LLP) method. LLP pads an input series of length L to match the length of complete series $L + T$ with a learnable linear layer. This can be formulated as:

$$\mathbf{X}_p = \mathbf{concat}(\mathbf{X}, \mathbf{W}^T\mathbf{X} + b),$$

where $\mathbf{X} \in \mathbb{R}^L$ represents the input series, $\mathbf{W} \in \mathbb{R}^{L \times T}$ and $b \in \mathbb{R}^T$ are the weight matrix and bias vector respectively, and $\mathbf{X}_p \in \mathbb{R}^{L+T}$ is the padded series.

By applying Learnable Linear Padding, we can obtain the spectrum of the input series with a frequency resolution that is aligned with that of the complete series. This effectively eliminates the frequency-resolution misalignment issue. As a result, it enables more effective extraction of characteristics from the frequency domain, ultimately contributing to enhanced forecasting performance.

4.2 Frequency Filter

After frequency domain transformation (via Frequency Resolution Alignment and Learnable Linear Padding), the model is tasked with learning global patterns but faces two challenges (Figure 5): (1) high-frequency noise introduces performance-degrading artifacts [14, 55], and (2) deep learning approaches are inclined to prioritize low-frequency information during training due to its typically larger magnitude [38], which can result in the loss of critical periodic

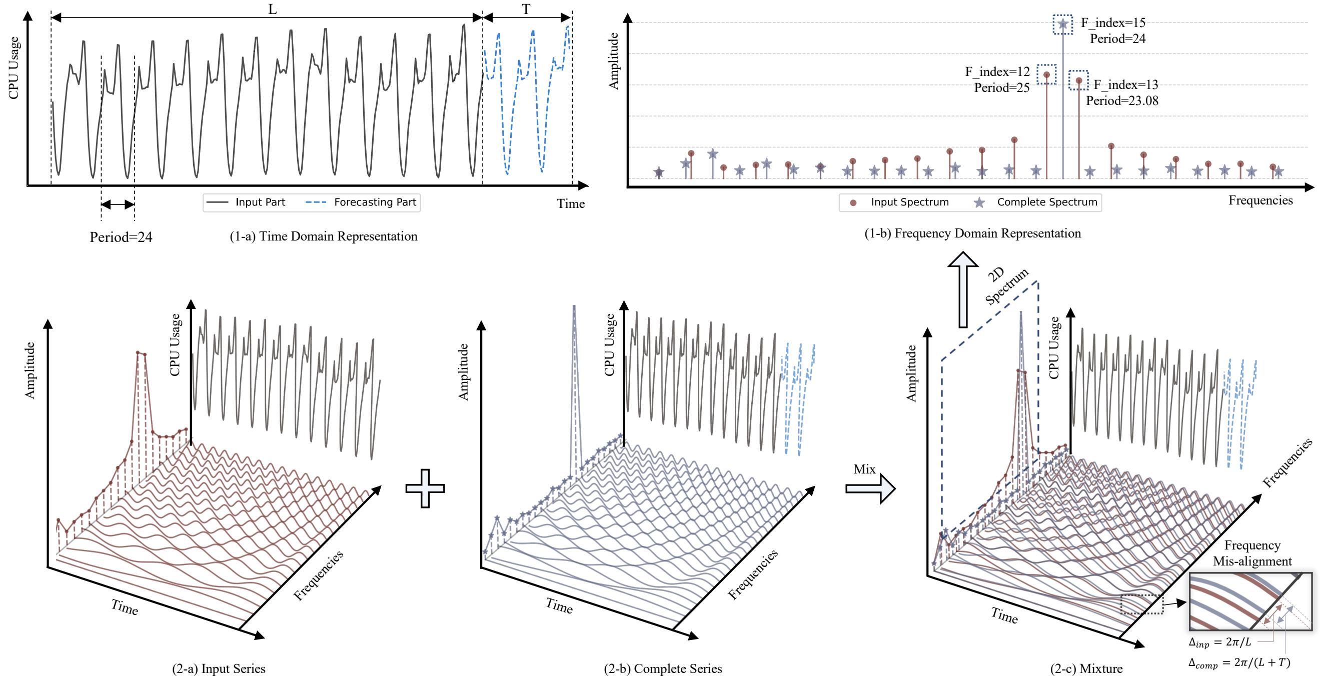


Figure 4: Frequency Resolution Mis-alignment. The original workload series is sourced from PaaS dataset. It could be observed that the spectrum of input series and complete series mis-align in resolution.

patterns. To address them, we design Low-Pass Filter (LPF) and High-Pass Filter (HPF) respectively.

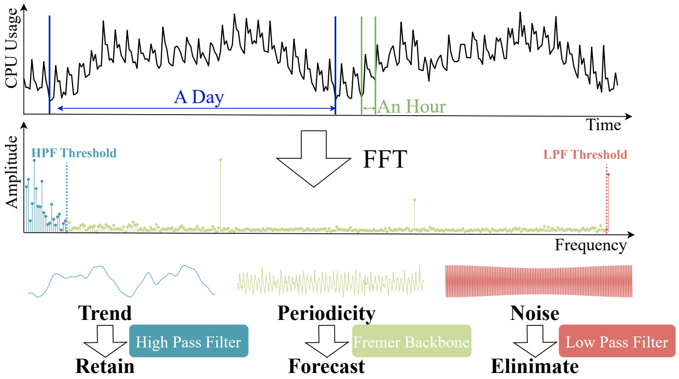


Figure 5: Frequency Filter is used to retain low-frequency trend information and eliminate high-frequency noise.

LPF aims to eliminate noise by zeroing frequencies above its threshold (the highest 1% of frequencies by default). For the low-frequency components, two key observations are noted: (1) the low-frequency portion typically contains trend information, such as the zero-frequency component, which captures the mean value of the sequence; (2) low-frequency components are a primary contributor to model overfitting, as the model prioritizes fitting these components first [38]. We need both to retain the trend information in the low-frequency part and to prevent the model from overfitting to it. Thus, we preserve the lowest 3% of frequencies (by default) but exclude them from `Fremer`'s backbone input to prevent overfitting, later concatenating these components with the backbone's outputs to generate final predictions.

4.3 Complex-valued Spectrum Attention

After padded with LLP, transposing to frequency domain with rFFT and applying the frequency filters, we obtain the input spectrum $\mathbf{F}_f \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1 - L_f}$, where L_f is the number of frequencies that are filtered out by the filters.

Firstly, we utilize Frequency Reversible Instance Norm (F-RIN) to process the input spectrum $\mathbf{F}_f$. While RevIN[18] was originally developed to address distribution shift in the time domain, we have found it to be effective in handling frequency domain spectra as well. This approach enables transforming the spectra of series with distinct global features into a similar distribution. To accomplish this, $\mathbf{F}_f$ is normalized by subtracting its mean and dividing it by its standard deviation. The standard deviation is calculated using $|\mathbf{F}_f| \in \mathbb{R}^{\lfloor \frac{L+T}{2} \rfloor + 1 - L_f}$. Specifically, we obtain the normed spectrum $\mathbf{F}_r$ by the following process:

$$\mathbf{F}_r = \frac{\mathbf{F}_f - \mathbf{mean}(\mathbf{F}_f)}{\mathbf{std}(|\mathbf{F}_f|)}. \quad (3)$$

Afterwards, we add them back at the end of the structure. Next, a trainable linear projection $\mathbf{W}_{proj} \in \mathbb{C}^{L' \times (\lfloor \frac{L+T}{2} \rfloor + 1 - L_f)}$ is employed to obtain $\mathbf{F}_c \in \mathbb{C}^{L'}$, which represents the linear combinations of $\mathbf{F}_r$.

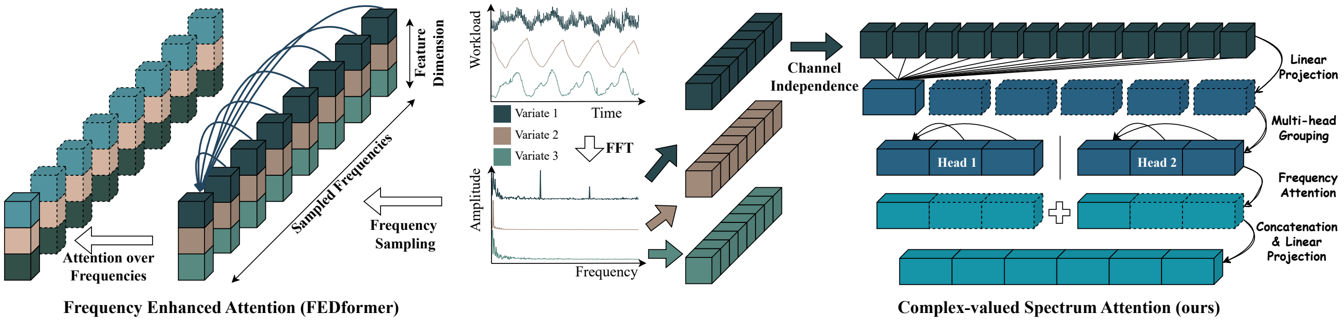


Figure 6: Comparison between Frequency Enhanced Attention (FEA) proposed by FEDformer [61]) and our proposed CSA.

Complex-valued Spectrum Attention. As a single frequency point is scarcely capable of carrying semantic meaning, we developed Complex-valued Spectrum Attention (CSA). CSA is designed to capture attention across frequency combinations, which enables it to more effectively capture the characteristics of the frequency domain. Specifically, We apply a modified multi-head attention mechanism rather than the original one in [48]. For each head $h = 1, 2, \ldots, H$, it projects the linear combined spectrum $\mathbf{F}_c \in \mathbb{C}^{L'}$ to dimension l across the spectrum dimension with trainable projections. Specifically, $\mathbf{Q}_h = \mathbf{F}_c^T \mathbf{W}_h^Q$, $\mathbf{K}_h = \mathbf{F}_c^T \mathbf{W}_h^K$, $\mathbf{V}_h = \mathbf{F}_c^T \mathbf{W}_h^V$, where $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{C}^{L' \times l}$ are learnable parameters. On each head we perform complex-valued Dot-Product Attention:

$$\mathbf{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \mathbf{Softmax}(|\mathbf{Q}_h \mathbf{K}_h^T|)\mathbf{V}_h. \tag{4}$$

Then the final output of Complex-valued Spectrum Attention is calculated as follows:

$$\begin{aligned} \mathrm{head}_h &= \mathbf{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h), \\ \mathrm{CSA}(\mathbf{F}_c) &= \mathbf{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_H)^T \mathbf{W}_O, \end{aligned} \tag{5}$$

where $\mathbf{W}_O \in \mathbb{C}^{hl \times L'}$ are learnable parameters. The inverted multi-head projection not only enables capturing relationships among frequency combinations from multiple perspective, but also reduces the complexity to $\frac{1}{H}$ of the original approach.

Figure 6 presents a detailed visualization comparing the Complex-valued Spectrum Attention (CSA) with the Frequency Enhanced Attention (FEA) proposed in FEDformer [61], both of which employ attention in the frequency domain representation. There exist two key distinctions between them:

- FEA employs a channel-mixing strategy, while CSA adopts a channel-independent approach. Since frequency domain representations primarily capture global patterns within a series—patterns unlikely to correlate across channels—we argue that channel independence is more effective for handling frequency domain data.
- FEA computes attention over sampled frequencies, while CSA computes attention over combinations of frequencies. Since a single frequency point is unlikely to carry semantic meaning, capturing attention across combinations of frequencies will yield greater benefits.

We follow the design of LayerNorm and FeedForward layers with residual connections in Transformer, with expanding to complex number field. Subsequently, following operations through a linear projection layer, a frequency-recovery process, and a Frequency Reversible Instance Norm (F-RIN) process, we obtain the output $\hat{\mathbf{F}} \in \mathbb{C}^{\lfloor \frac{L+T}{2} \rfloor + 1}$. We convert $\hat{\mathbf{F}}$ to time domain using inverse real-valued Fast Fourier Transform (irFFT), taking the last T points (the forecasting part) as the final output of `Fremer` $\hat{\mathbf{X}} \in \mathbb{R}^T$.

4.4 Complexity Analysis

The primary computational bottleneck of the Transformer-based model lies in the Attention mechanism. The dot-product Attention mechanism has a complexity of $O(L)$, where L is the input sequence length. To reduce this complexity, `Fremer` uses a series of designs, ultimately lowering it to $O(\frac{L'^2}{H})$, where L', H are the number of the frequency combinations and the Attention heads.

Firstly, according to the symmetry property of the Discrete Fourier Transform, the spectrum length in the frequency domain is $\frac{L}{2}$ for a sequence of length L transformed into the frequency domain. Considering the use of LLP and Frequency Filter, the actual length is $\frac{L+T}{2} - L_f$. Furthermore, based on the observation that frequency-domain information is concentrated and exists in combinations, we map frequency information to combinations to further reduces computational complexity. The sequence length is further reduced to L' (defaulting to $\frac{L}{5}$). The Attention formula is as follows:

$$\mathbf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{Softmax}(|\mathbf{Q}\mathbf{K}^T|)\mathbf{V},$$

where $\mathbf{Q} = \mathbf{F}_c^T \mathbf{W}^Q$, $\mathbf{K} = \mathbf{F}_c^T \mathbf{W}^K$, $\mathbf{V} = \mathbf{F}_c^T \mathbf{W}^V$ and $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{C}^{1 \times L'}$. Consequently, the computational complexity remains $O((L')^2)$. By applying the inverted multi-head projection, we decompose the original frequency combinations into H independent projection heads: $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$, for $h = 1 \ldots H$, and the attention computation for each head is then given by:

$$\mathbf{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \mathbf{Softmax}(|\mathbf{Q}_h \mathbf{K}_h^T|)\mathbf{V}_h,$$

where $\mathbf{Q}_h = \mathbf{F}_c^T \mathbf{W}_h^Q$, $\mathbf{K}_h = \mathbf{F}_c^T \mathbf{W}_h^K$, $\mathbf{V}_h = \mathbf{F}_c^T \mathbf{W}_h^V$ and $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h \in \mathbb{C}^{1 \times \frac{L'}{h}}$. The computational complexity will be $O(H \times (\frac{L'}{H})^2) = O(\frac{(L')^2}{H})$. Compared to the dot-product Attention, the computational complexity of the Complex-valued Spectrum Attention used in `Fremer` is reduced to approximately $\frac{1}{200}$ of the original.We further compare the attention complexity of `Fremer` with other Transformer-based forecasting models, as summarized in Table 1.

Table 1: Attention Complexity of Transformer-based Forecasting Models. Let L denote the input series length, L' the number of frequency combinations, H the number of attention heads, D the hidden dimension, P the patching stride, and L_f the number of sampled frequencies in FEDformer.

Model	Complexity	Model	Complexity
Fremer	$O(\frac{L'^2}{H})$	iTransformer	$O(D^2)$
Informer	$O(DL \log L)$	Fredformer	$O(\frac{D^2}{P} L)$
PatchTST	$O(\frac{D}{P^2} L^2)$	FEDformer	$O(DL_f^2)$
Transformer	$O(DL^2)$	Crossformer	$O(\frac{D}{P^2} L^2)$

5 EXPERIMENTS

In this section, we conduct extensive experiments to address the following questions:

RQ1 How does the performance of Fremer compare with SOTA time series forecasting models in workload forecasting?

RQ2 Can the key components within Fremer be identified as significant contributors to its performance?

RQ3 How well does Fremer deal with the historical data sparsity problem?

RQ4 Can Fremer well balance the efficiency and effectiveness compared to other models?

RQ5 Besides workload forecasting, can Fremer demonstrate competitive performance in other forecasting scenarios?

RQ6 In the cloud service scenarios, can Fremer help improve the system performance, such as achieving lower latency?

5.1 Experiment Settings

Datasets. We present four high-caliber workload datasets sourced from diverse cloud services within ByteDance, as detailed below:

FaaS (Function as a Service). ByteFaaS provides a highly scalable and efficient way to execute discrete functions. The dataset includes the QPS (Queries Per Second) data of function instances.

IaaS (Infrastructure as a Service). IaaS supplies users with foundational computing resources such as virtual machines, storage, etc.. The dataset records the CPU usage data of virtual machines.

PaaS (Platform as a Service). PaaS offers a comprehensive development and deployment environment. The dataset contains CPU usage information of individual services measured in milli-cores.

RDS (Relational Database Service). RDS is designed for efficient management and storage of structured data. The dataset includes QPS data from MySQL instances.

In addition, we also incorporate 3 public datasets from Materna-Workload-Traces[2] [47]. These datasets are provided by GWA-T-13 MATERNA and originate from a German-based distributed cloud data center. They comprise three distinct traces, with each trace containing 520, 527, and 547 virtual machines (VMs), respectively. We choose the metric "CPU usage" as workload for forecasting and VMs exhibiting a missing rate (proportion of missing time points over total length) exceeding 2% are filtered out. We also summarize the information of datasets utilized in this study in Table 2.

Table 2: Statistics of workload datasets.

Dataset	# of Instances	Lengths	Frequency	Start Date	End Date
FaaS	226	2305	10-min	2022-04-02	2022-04-18
IaaS	93	3456	10-min	2023-06-30	2023-07-24
PaaS	426	7776	10-min	2024-09-01	2024-10-24
RDS	1113	6624	10-min	2024-08-16	2024-09-30
MT-1	413	8352	5-min	2015-11-05	2015-12-03
MT-2	402	8928	5-min	2015-12-04	2016-01-03
MT-3	371	10368	5-min	2016-01-04	2016-02-08

Baselines. We select representative models as baselines, including Fredformer [38], PDF [9], iTransformer [30], FEDformer [61], Crossformer [58], Informer [59], PatchTST [35], DLinear [57], NLinear [57], FITS [53], MICN [49], TimesNet [51], and FECAM [14].

Evaluation Platform. To ensure a fair and unbiased comparison, we adopt the Time Series Forecasting Benchmark (TFB)[39] as our evaluation platform. TFB includes implementations of all baseline models, and we retain the default settings for each model as provided by the platform. The datasets are systematically organized in the TFB format and divided into training, validation, and test sets using a 7:1:2 ratio based on the time span.

Evaluation Metrics. Due to the magnitude differences across datasets (e.g., CPU utilization ranges from $[0, 100]$, while QPS can reach up to $10^6 \sim 10^7$), computing metrics like mean squared error (MSE) and mean absolute error (MAE) on raw data is dataset-dependent and lacks comparability across different datasets. Therefore, we first normalize the data to calculate MSE_{norm} and MAE_{norm}. Also, since symmetric mean absolute percentage error (SMAPE), which measures the relative magnitude of prediction errors, is not affected by the magnitude of the data, we compute it on raw data to comprehensively evaluate prediction errors. The specific calculating process of the test phase are as follows:

$$
\begin{aligned}
(\mathbf{X}^{(n)}_{\text{norm}}, \mathbf{Y}^{(n)}_{\text{norm}}) &= \text{Normalization}((\mathbf{X}^{(n)}, \mathbf{Y}^{(n)})),\\
\hat{\mathbf{Y}}^{(n)}_{\text{norm}} &= \text{Fremer}(\mathbf{X}^{(n)}_{\text{norm}}),\\
\text{MSE}_{\text{norm}} &= \frac{1}{NH}\sum_{n=0}^{N-1} ||\hat{\mathbf{Y}}^{(n)}_{\text{norm}} - \mathbf{Y}^{(n)}_{\text{norm}}||_2,\\
\text{MAE}_{\text{norm}} &= \frac{1}{NH}\sum_{n=0}^{N-1} ||\hat{\mathbf{Y}}^{(n)}_{\text{norm}} - \mathbf{Y}^{(n)}_{\text{norm}}||_1,\\
\hat{\mathbf{Y}}^{(n)} &= \text{Denormalization}(\hat{\mathbf{Y}}^{(n)}_{\text{norm}}),\\
\text{SMAPE} &= \frac{100\%}{NH}\sum_{n=1}^{N}\sum_{i=1}^{H} \frac{2|\hat{\mathbf{Y}}^{(n)}_i - \mathbf{Y}^{(n)}_i|}{|\hat{\mathbf{Y}}^{(n)}_i| + |\mathbf{Y}^{(n)}_i|},
\end{aligned}
$$

where N denotes the size of the test set, H denotes the forecasting horizon, $(\mathbf{X}^{(n)} = [\mathbf{X}^{(n)}_1, \dots, \mathbf{X}^{(n)}_L], \mathbf{Y}^{(n)} = [\mathbf{Y}^{(n)}_1, \dots, \mathbf{Y}^{(n)}_H])$ denotes the n-th test sample, and $\hat{\mathbf{Y}}^{(n)}_{\text{norm}}, \hat{\mathbf{Y}}^{(n)} \in \mathbb{R}^N$ denote the normalized and denormalized forecasts, respectively. We use the sequence mean and standard deviation of the training set for data normalization and denormalization, thus avoiding test data leakage.

[2] https://www.kaggle.com/datasets/kpiyush04/maternaworkloadtraces

Table 3: Results of Workload forecasting. The best results are bold and the second-best results are underlined. "Seasonality" and "Correlation" are statistical characteristics of the dataset calculated per TFB [39].

Architectures				Transformer								MLP			CNN		Attention
Datasets	Seasonality	Correlation	Metrics	Fremer	Fredformer	PDF	iTransformer	PatchTST	Crossformer	FEDformer	Informer	FITS	DLinear	NLinear	MICN	TimesNet	FECAM
PaaS	0.974	0.897	MSE	**0.062**	0.072	0.065	0.072	0.082	0.220	2.262	0.249	0.066	0.067	0.068	0.079	0.553	0.066
			MAE	**0.137**	0.161	0.146	0.155	0.183	0.316	1.191	0.355	0.145	0.146	0.149	0.167	0.563	0.142
			SMAPE(%)	**5.329**	6.527	5.878	6.461	7.928	13.982	49.226	15.655	5.664	5.739	5.943	6.971	25.170	5.656
FaaS	0.909	0.740	MSE	**0.289**	0.431	0.319	0.327	0.324	1.123	1.499	0.900	0.430	0.365	0.342	0.383	0.631	0.397
			MAE	**0.314**	0.402	0.341	0.346	0.350	0.849	0.960	0.684	0.472	0.391	0.355	0.392	0.570	0.413
			SMAPE(%)	**9.240**	11.6431	9.917	10.207	10.626	24.832	27.784	20.180	14.661	11.264	10.658	11.450	17.120	12.095
RDS	0.879	0.707	MSE	**1.292**	1.520	1.489	1.333	1.774	5.636	3.853	3.967	1.481	1.625	1.445	2.034	3.555	1.585
			MAE	**0.364**	0.407	0.396	0.388	0.431	0.897	0.962	0.766	0.405	0.420	0.415	0.468	0.770	0.389
			SMAPE(%)	**8.663**	9.861	9.702	9.145	10.714	25.207	27.254	20.973	9.828	9.810	10.147	10.867	21.4237	8.760
IaaS	0.819	0.742	MSE	**0.708**	0.744	0.755	0.770	0.746	1.106	0.999	0.936	0.746	0.753	0.818	0.785	1.099	0.762
			MAE	**0.556**	0.580	0.593	0.610	0.582	0.782	0.746	0.689	0.588	0.589	0.618	0.609	0.763	0.591
			SMAPE(%)	**16.249**	16.933	17.025	16.705	17.083	22.521	21.741	19.990	17.199	16.944	17.870	17.806	21.872	17.132
MT1	0.635	0.731	MSE	15.408	15.992	15.250	**15.205**	15.654	15.654	16.092	16.988	15.707	15.766	16.076	18.226	16.173	15.368
			MAE	0.433	0.466	0.447	0.435	0.474	0.531	0.750	0.641	0.485	0.479	0.484	0.845	0.636	**0.415**
			SMAPE(%)	**18.828**	20.116	19.127	18.829	21.024	24.734	38.943	26.474	20.269	21.636	20.696	33.932	27.567	19.152
MT3	0.628	0.723	MSE	**28.062**	32.778	28.224	28.937	28.476	72.847	35.837	36.365	28.391	33.205	29.090	47.147	35.895	50.023
			MAE	0.699	0.777	**0.687**	0.713	0.718	1.120	0.965	1.074	0.761	0.856	0.851	1.079	0.946	0.852
			SMAPE(%)	**21.411**	23.620	22.352	21.886	22.645	25.859	34.105	33.722	22.911	24.333	29.269	29.703	28.225	22.220
MT2	0.618	0.714	MSE	6.543	6.618	6.481	6.559	**6.467**	6.754	6.907	6.969	6.720	6.678	6.678	6.758	6.848	6.554
			MAE	**0.287**	0.309	0.296	0.290	0.302	0.356	0.532	0.447	0.352	0.332	0.305	0.401	0.433	0.300
			SMAPE(%)	**14.360**	16.038	14.629	14.927	15.470	19.329	31.562	20.659	16.064	17.628	14.554	21.145	20.676	16.614

Implementation Details. We implement Fremer and all baseline models utilizing the PyTorch framework [37], and conduct the training process on NVIDIA A100-SXM 80GB GPUs. The ADAM optimizer [19] is selected, with an initial learning rate of $1e^{-3}$, and the MSE-Loss is designated as the optimization objective. Throughout all experiments, the batch size is consistently set to 32, and the number of training epochs is fixed at 20. Regarding the model hyper-parameters, such as the model dimension and the number of encoder layers, for the baseline models, we adopt the default settings provided within TFB. For Fremer, a limited hyper-parameter search is performed, with the primary focus on the frequency filter threshold. In the case of training hyper-parameters, including batch size and learning rate, a uniform configuration is applied across all models. This standardized implementation approach ensures the reproducibility and comparability of the experimental results, facilitating a more accurate assessment of the performance of Fremer and the baseline models within the research context.

5.2 RQ1: Workload Forecasting Result

In this section, an extensive assessment of the forecasting performance of Fremer in conjunction with other baseline models is conducted across the seven aforementioned workload datasets. The input window is configured as 5 days, corresponding to 1440 data points for datasets with a 5-minute granularity and 720 points for those with a 10-minute granularity. The forecasting horizon is set to 1 day, equivalent to 288 points for 5-minute granularity datasets and 144 points for 10-minute granularity datasets.

As the results shown in Table 3, Fremer achieve the best performance on most datasets. Specifically, in comparison to the best performing baseline models within each dataset, Fremer attains an average enhancement in forecasting accuracy of 2.9% in terms of MSE, 2.5% in MAE, and 3.5% in SMAPE. These outcomes affirm the superiority of Fremer for workload forecasting. Additionally, it can be observed that for models that explicitly model the correlative relationships among workload series of distinct computing instances, their forecasting performance is relatively inferior to those models with channel independence. This phenomenon can be attributed to the relatively autonomous nature of workloads across distinct instances within these datasets, rendering the modeling of such correlative relationships ineffective.

We present visualizations of the forecasting results of Fremer in comparison with some high-performing baseline models, as depicted in Figure 7. From these visualizations, it can be clearly observed that Fremer produces more accurate forecasts compared to other models. FaaS data shows complicated periodic pattern and Fremer captures the pattern accurately. RDS data shows local period shift and Fremer produce accurate predictions.

5.3 RQ2: Components Analysis

5.3.1 Ablation Study. We conduct the ablation study on the Fremer's key components, the Learnable Linear Padding (LLP), the Complex-valued Spectrum Attention (CSA), the Frequency Filter (F-Filter) and the Frequency-Reversible Instance Normalization (F-RIN) to analyze their contributions, as shown in Figure 8. Specifically, the notations "w/o LLP", "w/o F-Filter", and "w/o F-RIN" designate Fremer

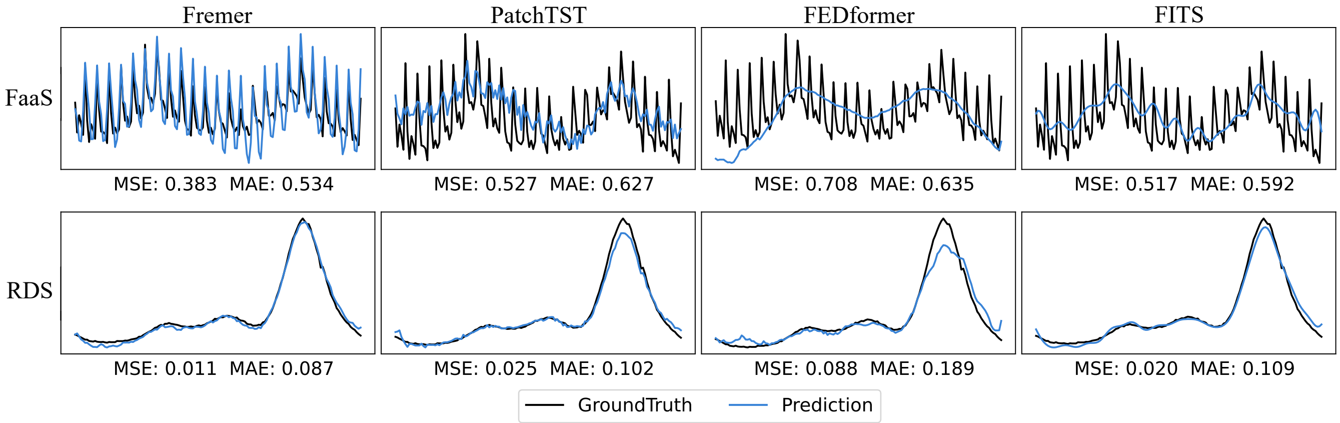


Figure 7: Visualization of Workload Forecasting Results on Distinct Datasets.

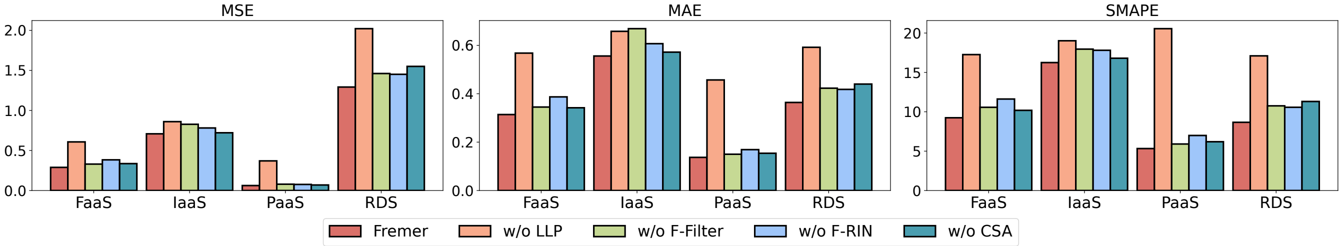


Figure 8: Ablation Study Results. We highlight the contributions of four key modules of Fremer: Learnable Linear Padding (LLP), Frequency Filter (F-Filter), Frequency Instance Normalization (F-RIN), and Complex-valued Spectrum Attention (CSA).

without the corresponding modules, while "w/o CSA" represents Fremer implemented with a frequency attention mechanism computed across individual frequency components. We observe that each component plays a crucial role in enhancing the forecasting performance of Fremer. Specifically, removing the LLP module leads to a significant drop in performance, highlighting the critical importance of frequency alignment. By effectively using the LLP to align frequency resolutions, Fremer achieves higher forecasting accuracy. Besides, when the dataset, such as IaaS, has strong noise, removing F-Filter will incur the remarkable performance degradation.Similarly, the absence of F-RIN and CSA will also cause the performance degradation, which underscores the effectiveness of applying RevIN to frequency domain representations and performing Attention on frequency combinations.

5.3.2 Frequency Filters. We also investigate the impact of frequency filter thresholds on Fremer. The results are presented in Figure 9. From Figure 9(a), it can be observed that when applying the High-Pass Filter (HPF), even with a small threshold, the training loss increases while the validation and test sets decrease. This indicates that without the HPF, Fremer has a tendency to overfit on the training set. Since the low-frequency part has a relatively larger influence on the forecasting results, simply retaining the low-frequency part can effectively address the overfitting problem. This provides a convenient way to control the generalizability of Fremer for different workloads. From Figure 9(b), it can be seen that removing the highest frequency noise reduces the training, validation, and testing losses. Only the highest-frequency noise significantly affects model performance, and carefully removing it can help the model learn better while retaining the useful information.

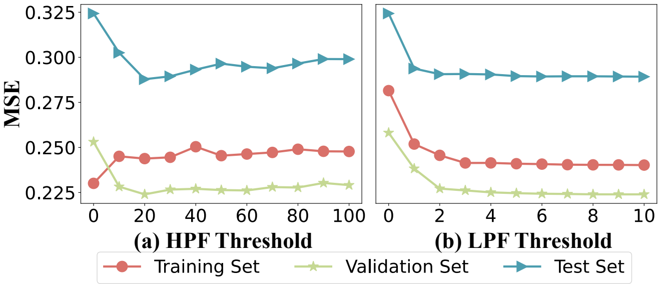
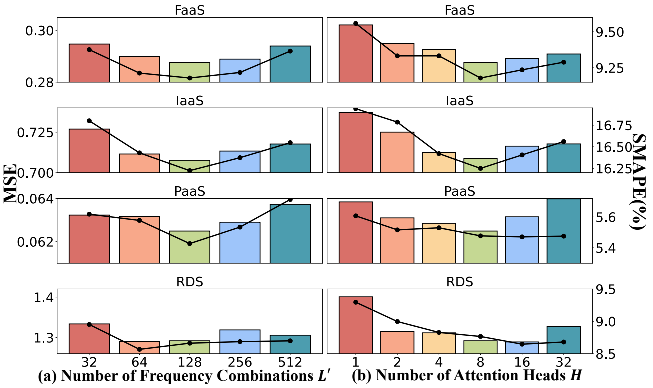


Figure 9: Frequency Filter Analysis Results. Figure (a) and (b) respectively show the impact of different HPF and LPF thresholds on training/validation/testing losses.

5.3.3 Parameter Sensitivity. We analyze the impact of key parameters on model performance, as shown in Figure 10. Figure 10(a) shows the impact of different frequency combination numbers L'. As can be seen, a larger L' does not always lead to better results. When L' is approximately $\frac{L}{5}$ (for an input length of $L = 720$, $L' = 144$), Fremer achieves optimal performance. This aligns with our discussion in Section 4.4 and explains the efficiency of Fremer, as it can capture long sequence periodic features with fewer frequency combinations. Figure 10(b) shows the impact of different attention head numbers h. Overall, as h increases, the performance of Fremer improves, but with diminishing returns. Excessively large h can even increase error. Considering the balance between efficiency and effectiveness, $h = 8$ is a reasonable choice.



(a) Number of Frequency Combinations L' (b) Number of Attention Heads H

Figure 10: The Parameter Analysis. The bar chart indicates MSE, and the line chart indicates SMAPE. Figure (a) shows the impact of the frequency combination number L', and (b) shows the impact of the attention head number H.

5.3.4 *Input Length L and Forecast Horizon T.*

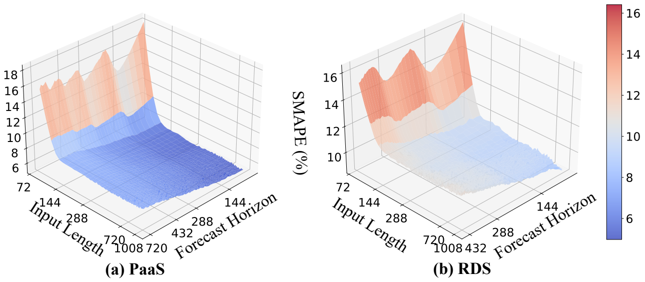
We evaluate Fremer on the PaaS and RDS datasets across varying input lengths ($L \in [36, 1008]$) and forecasting horizons ($H \in [18, 720]$, spanning hours to days), with results in Figure 11. They show that once L exceeds 144 (i.e., the input includes a day's data), Fremer 's prediction error drops sharply, highlighting its ability to capture data periodicity. The impact of output length on the model's effectiveness is influenced by the input length. When the input length is short ($L < 100$), the model's effectiveness is better when the output length is a multiple of the period. After the input length increases, the output results become more stable. When the input length is 720, the SMPAE for an output length of 720 increases by only 40% compared to that for an output length of 72.

5.3.5 *Multi-Head Attention Mechanism.*

We visualize and analyze the multi-head CSA when forecasting FaaS workload, as shown in the figure 12. By embedding frequency points to combinations, CSA represents periodic information with lower complexity. The Multi-head mechanism further enhances the capture of different periodic elements. In the attention score heatmap, row n shows the attention scores of frequency combination m to others, and column n shows the attention scores received by combination n from others. In Head 2 and Head 6, the two most attended combinations are highlighted, showing their focus on different frequency information: the left on the hourly period, the right on the sub-hourly period.

5.4 RQ3: Generalizability Test

To comprehensively assess the generalizability of Fremer, under the two experimental setups, the intra- and cross-dataset transfer, we validate the performance of Fremer. Note that models requiring fixed channels are unsuitable for transfer learning, as the number of channels in training and testing datasets often differs.



(a) PaaS (b) RDS

Figure 11: The sensitivity to L and T of Fremer. The three axes respectively represent the input length L, the forecast horizon T, and the SMAPE.

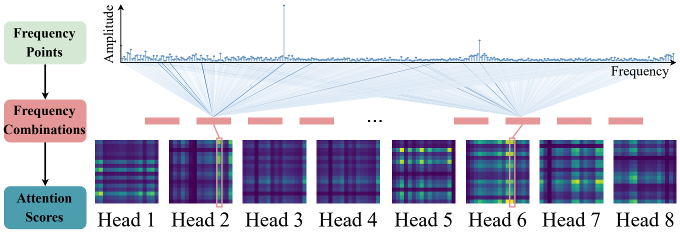


Figure 12: Multi-head CSA. The figure shows the mapping from frequency points (shown in the top spectrum) to frequency combinations (indicated by middle red rectangles). The thicker the blue lines between them, the higher the weights. The bottom part is a heatmap of attention scores, where brightness indicates the score's magnitude.

Table 4: Results of Intra-Dataset Transfer Forecasting.

		Fremer	PDF	iTrans.	PatchTST	FITS	DLinear	NLinear
FaaS	MSE	**0.325**	0.335	0.355	0.337	0.349	0.320	0.348
	SMAPE(%)	**9.645**	9.866	10.331	9.834	10.475	9.782	10.161
IaaS	MSE	**0.707**	0.710	0.714	0.722	0.708	0.711	0.720
	SMAPE(%)	15.988	16.023	16.270	16.003	16.088	**15.979**	16.043
RDS	MSE	**0.351**	**0.351**	0.360	0.388	0.371	0.361	0.368
	SMAPE(%)	**9.688**	9.695	10.232	10.699	10.652	10.384	10.573
PaaS	MSE	**0.045**	0.048	0.051	0.051	0.052	0.050	0.052
	SMAPE(%)	**6.503**	6.619	6.582	6.659	8.723	8.114	7.687
MT1	MSE	6.856	7.080	7.033	6.869	**6.798**	6.874	7.448
	SMAPE(%)	**23.244**	23.595	24.207	25.851	25.419	26.590	37.013
MT2	MSE	**6.782**	6.982	6.835	7.074	7.093	7.017	7.177
	SMAPE(%)	**17.320**	17.500	18.319	20.602	18.810	18.151	17.534
MT3	MSE	15.378	15.221	15.258	15.115	15.012	**14.766**	18.234
	SMAPE(%)	**25.052**	25.750	25.455	25.899	26.482	25.801	31.571

5.4.1 *Intra-Dataset Transfer.*

In this experimental setup, models are trained on a dataset's training instances and evaluated on the remaining test set. By default, the ratio of the number of training instances to test instances is set at 8:2. As can be seen from the results presented in Table 4, Fremer achieves the best performance.

Table 5: Results of Cross-Dataset Transfer Forecasting.

Source: PaaS		Fremer	PDF	iTrans.	PatchTST	FITS	DLinear	NLinear
FaaS	MSE	**0.308**	0.361	0.377	0.312	0.322	0.312	0.333
	SMAPE(%)	**9.204**	9.932	11.233	9.383	9.780	9.436	10.014
IaaS	MSE	**0.763**	0.810	1.027	0.798	0.780	0.764	0.966
	SMAPE(%)	**17.360**	18.254	21.469	18.021	17.649	17.478	19.781
RDS	MSE	**1.539**	1.643	1.898	1.566	1.856	1.607	1.516
	SMAPE(%)	**9.634**	10.539	13.474	10.007	10.268	9.965	10.854

Source: RDS		Fremer	PDF	iTrans.	PatchTST	FITS	DLinear	NLinear
FaaS	MSE	**0.311**	0.317	0.313	0.328	0.323	0.316	0.326
	SMAPE(%)	**9.095**	9.190	9.704	9.507	10.151	9.405	9.475
IaaS	MSE	**0.729**	0.736	0.891	0.788	0.760	0.756	0.810
	SMAPE(%)	**16.696**	16.751	19.935	17.838	17.294	16.958	17.730
PaaS	MSE	**0.063**	0.064	0.111	0.068	0.085	0.071	0.066
	SMAPE(%)	**5.628**	5.648	12.148	6.461	8.929	6.606	6.222

Table 6: Results of Efficiency Test on Transformer-based Models."Training" means training time (ms) per iteration, "Inference" means inference time (ms) per iteration, and "Parameters" means parameter count (M).

Models		Fremer	Fred.	PDF	iTrans.	PatchTST	Cross.	FED.	In.
IaaS	Training	61.67	214.56	126.01	91.92	346.47	236.83	437.42	82.48
	Inference	34.39	92.44	42.97	36.87	47.77	91.56	214.45	62.39
	Parameter	0.57	111.13	6.04	0.76	6.85	11.99	14.12	0.60
	SMAPE(%)	16.25	16.93	17.03	16.71	17.08	22.52	21.74	19.99
RDS	Training	105.23	5831.20	601.12	270.03	560.41	427.93	424.05	85.36
	Inference	27.87	3006.61	235.19	103.29	32.09	100.09	186.10	50.46
	Parameter	0.59	113.46	7.77	0.77	4.38	5.93	2.28	4.04
	SMAPE(%)	8.66	9.86	9.70	9.15	10.71	25.21	27.25	20.97

This remarkable result clearly demonstrates the strong generalizability of Fremer when making predictions on workload series from unseen instances within the same dataset. Fremer can effectively capture the underlying patterns and characteristics present in the training data and apply them to new, unencountered data segments within the same dataset, showcasing its adaptability and robustness.

5.4.2 Cross-Dataset Transfer. In this experimental setup, models are trained on the training set of all instances from one dataset and evaluated on the test set of all instances from another dataset. As shown in Table 5, Fremer consistently achieves top performance, demonstrating exceptional zero-shot forecasting ability. This highlights Fremer 's exceptional zero-shot forecasting ability and underscores its potential as a foundational backbone for large-scale workload forecasting models. Fremer 's ability to generalize across datasets, capturing both commonalities and unique features of workload data, is a critical characteristic for building more comprehensive and powerful forecasting models.

5.5 RQ4: Efficiency-Effectiveness Analysis

In this section, we evaluate the efficiency of Fremer in workload forecasting experiments, comparing it directly with other Transformer-based models. For a thorough analysis, we chose two datasets of different sizes: the smaller IaaS dataset with 93 instances, and the larger RDS dataset with 1113 instances. The efficiency evaluation results are summarized in Table 6. From the table, it is clear that Fremer significantly outperforms other Transformer-based methods in terms of efficiency. Compared to the state-of-the-art model PatchTST, Fremer achieves notable improvements. For the IaaS dataset, it reduces training time by 82.2%, inference time by 28.0%, and parameter size by 91.7%. Similarly, for the RDS dataset, it cuts training time by 81.2%, inference time by 13.1%, and parameter size by 86.5%. Frequency domain methods (e.g., FredFormer, PDF, FEDformer) are often less efficient than time domain methods due to FFT-related overhead. However, Fremer, with its efficient design, achieves the best balance between efficiency and effectiveness. These improvements are attributed to Fremer 's lightweight components: the Complex-valued Spectrum Attention (CSA) mechanism and the frequency filter. The CSA efficiently extracts relevant information from the frequency domain while maintaining computational simplicity. The frequency filter optimizes data flow by selectively processing frequency components, minimizing unnecessary computational overhead. Moreover, the results underscore Fremer 's advantage: superior forecasting performance and exceptional efficiency. This combination is crucial for real-world applications, where accurate and efficient workload forecasting is essential. Fremer thus emerges as a robust solution for addressing challenges across diverse industrial settings.

5.6 RQ5: Extended to General Forecasting

While designed for workload forecasting, Fremer demonstrates strong cross-domain performance on diverse domains. To explore this further, we select widely-used TFB datasets [39] including traffic and electricity prediction to test Fremer's general forecasting ability. As Table 7 shows, Fremer outperforms all baselines on periodic datasets (Traffic, Electricity, PEMS04) across four horizons 96,192,336,720, leveraging its frequency-domain design to capture global dependencies. This highlights Fremer's versatility for periodic forecasting tasks from traffic management to energy planning.

5.7 RQ6: Predictive Auto-Scaling

In this section, we perform proactive (predictive) Horizontal Pod Autoscaler (HPA) scaling simulation tests on a Kubernetes cluster. For the forecasting model, a 5-day historical window is employed to predict the next-day workload. Based on this predicted value, a 24-hour simulation test is carried out to assess the quality of service and resource overhead when applying different forecasting models to Kubernetes HPA's proactive auto-scaling. The results of the simulation experiment are presented in Table 8. These include delays at different quantiles (for evaluating service quality) and the average and maximum number of Pods (for evaluating resource consumption). Here, Naïve HPA denotes the native passive (responsive) HPA in Kubernetes. Ideal represents the HPA that

Table 7: Multivariate Forecasting Results on Datasets of Various Domains.

Models				Fremer		Fredformer		PDF		iTransformer		PatchTST		Crossformer		FEDformer		Informer		DLinear		NLinear		MICN		FECAM	
	Dataset	Season.	Corr.	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Strong Periodicity	Electricity	0.945	0.802	**0.256**	**0.161**	0.279	0.181	0.260	0.164	0.270	0.194	0.261	0.163	0.279	0.181	0.324	0.211	0.364	0.265	0.264	0.167	0.261	0.169	0.288	0.179	0.332	0.238
	Solar	0.919	0.753	**0.257**	**0.195**	0.287	0.226	0.264	0.200	0.262	0.202	0.294	0.207	0.365	0.330	0.482	0.421	0.397	0.380	0.309	0.247	0.272	0.255	0.296	0.248	0.315	0.261
	Traffic	0.880	0.813	**0.273**	**0.389**	0.339	0.517	0.279	0.399	0.281	0.397	0.283	0.405	0.284	0.523	0.377	0.615	0.414	0.752	0.295	0.434	0.290	0.433	0.308	0.534	0.445	0.715
	PEMS04	0.854	0.797	**0.241**	**0.142**	0.433	0.361	0.305	0.205	0.261	0.156	0.303	0.191	0.269	0.168	0.709	0.867	0.338	0.236	0.332	0.243	0.330	0.255	0.445	0.388	0.443	0.369
	PEMS08	0.850	0.807	**0.265**	0.288	0.486	0.582	0.326	0.373	0.271	0.298	0.306	0.213	0.273	**0.176**	0.706	0.876	0.402	0.350	0.363	0.298	0.360	0.322	0.472	0.436	0.493	0.620
Weak Periodicity	ZafNoo	0.757	0.598	0.464	0.522	0.467	0.595	0.453	0.515	0.456	0.523	0.465	0.511	0.455	**0.494**	0.499	0.578	0.602	0.744	**0.451**	0.496	0.458	0.522	0.452	0.495	0.520	0.717
	ETTh1	0.730	0.630	0.446	0.435	0.438	0.449	0.426	**0.407**	0.448	0.439	0.428	0.411	0.467	0.453	0.454	0.432	0.583	0.731	0.432	0.419	**0.421**	0.410	0.449	0.423	0.560	0.674
	AQShunyi	0.720	0.612	0.522	0.723	0.531	0.819	0.507	0.703	**0.503**	0.706	0.509	0.705	0.504	**0.694**	0.546	0.763	0.545	0.782	0.522	0.706	0.514	0.713	0.534	0.735	0.549	0.775
	Weather	0.652	0.663	0.284	0.241	0.272	0.244	0.263	0.227	0.270	0.232	**0.262**	**0.225**	0.294	0.235	0.351	0.306	0.323	0.300	0.289	0.239	0.281	0.249	0.288	0.238	0.305	0.260
	PEMS-BAY	0.618	0.842	0.375	**0.577**	0.454	0.759	0.399	0.688	0.381	0.583	0.398	0.663	**0.374**	0.590	0.598	0.959	0.460	0.899	0.443	0.719	0.445	0.748	0.476	0.854	0.544	0.959
	METR-LA	0.490	0.778	0.733	1.274	0.734	1.484	0.728	1.275	0.720	1.354	0.704	1.250	**0.695**	1.357	0.865	1.671	0.724	1.606	0.727	**1.203**	0.751	1.314	0.727	1.373	0.753	1.288

utilizes real values instead of predicted values for proactive (predictive) scaling, and it represents, to some extent, the performance upper-bound of predictive scaling. Ave-Lat represents the average latency, x-Lat represents the x-quantile latency, both measured in seconds (s). Timeout Rate denotes the number of requests that exceeds the timeout threshold, which is set to 10s in this experiment. AvePod represents the average number of Pods consumed. We utilize the workload series associated with a function instance in the ByteDance FaaS service for testing. Its workload data is also part of the FaaS dataset used to evaluate the workload-forecasting performance in section 5.2. Specifically, we record the workload and replay it from a client. The scaling strategy adheres to the original HPA mechanism, and is based on the assumption that the workload volume has a linear relationship with resource consumption.

Table 8: Kubernetes HPA Test Results

Models	Ave-Lat(s)	99.9-Lat(s)	99-Lat(s)	90-Lat(s)	Timeout Rate	AvePod
PatchTST	1.017	10.0	3.644	1.651	0.132%	22.479
DLinear	1.081	10.0	4.108	1.779	0.22%	19.25
FITS	1.05	10.0	4.101	1.704	0.279%	21.889
Naïve HPA	0.996	10.0	3.764	1.56	0.386%	29.181
Ideal	0.789	3.513	2.063	1.206	0.026%	21.382
Fremer	0.826	10.0	2.292	1.261	0.102%	21.951

Through a comprehensive examination and analysis of the experimental results, we conclude that the Fremer model demonstrates exceptional efficacy in forecasting future workloads. It accurately identifies trends in workload variations, thereby providing robust support for enhancing Quality of Service (QoS). This improvement is particularly evident in delay metrics across diverse quantiles. The Fremer model significantly outperforms other proactive autoscaling strategies based on alternative forecasting models. In practical applications, this implies that employing the Fremer model for workload prediction can more effectively ensure the smooth operation and stability of services, reduce user waiting times, and enhance the overall user experience.

Additionally, Fremer achieves remarkable results in optimizing resource utilization, as clearly reflected in the fluctuations of the average Pod count. For instance, comparative data analysis reveals that the Fremer model reduces average latency by 18.78% compared to the PatchTST model, while utilizing 2.35% fewer Pods on average. This indicates that the Fremer model can maintain service quality while using computing resources more efficiently, minimizing resource waste, and reducing operational costs for enterprises.

However, we also note that proactive HPAs guided by other forecasting models exhibit good performance in specific metrics. For example, the DLinear-based HPA achieves the lowest Pod usage. Nevertheless, its average latency is the highest among the compared models. Through detailed analysis, we attribute this to the DLinear's forecasts being significantly lower than the actual workload during certain periods. Consequently, the replica number recommendations derived from its predictions are insufficient to handle the actual workload, resulting in increased latency.

6 CONCLUSION

We propose Fremer, an efficient and effective deep forecasting model addressing critical challenges in cloud service workload forecasting. By utilizing frequency domain representations, Fremer balances accuracy, efficiency, and generalizability, meeting modern cloud requirements. Its innovations—Learnable Linear Padding, Frequency Filters, and Complex-valued Spectrum Attention—enable superior performance over state-of-the-art models with reduced computational costs. We also release four large-scale, high-quality datasets collected from ByteDance's cloud services, covering thousands of computing instances over 1–2 months, providing robust resources for research and benchmarking. Fremer addresses Transformer limitations and demonstrates strong generalizability, advancing cloud service forecasting. Future work will focus on scalable models for large cloud systems.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China [2024YFF0617700], the National Natural Science Foundation of China [U23A20309, 62272302], and the Bytedance Research Project [CT20241217115379].

REFERENCES

[1] Yanal Alahmad, Tariq Daradkeh, and Anjali Agarwal. 2021. Proactive failure-aware task scheduling framework for cloud computing. *IEEE Access* 9 (2021), 106152–106168.

[2] Alexandru-Florian Antonescu and Torsten Braun. 2016. Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications. *Future Generation computer systems* 54 (2016), 260–273.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[4] Masoud Barati and Saeed Sharifian. 2015. A hybrid heuristic-based tuned support vector regression model for cloud load prediction. *The Journal of Supercomputing* 71 (2015), 4235–4259.

[5] George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 17, 2 (1968), 91–109.

[6] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. 2014. Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE transactions on cloud computing* 3, 4 (2014), 449–458.

[7] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. 2022. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. arXiv:2201.12886 [cs.LG]

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[9] Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. 2024. Periodicity decoupling framework for long-term series forecasting. In *International Conference on Learning Representations*.

[10] Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang Bernie Wang, Mu Li, and Dit-Yan Yeung. 2022. Earthformer: Exploring space-time transformers for earth system forecasting. *Advances in Neural Information Processing Systems* 35 (2022), 25390–25403.

[11] Ehsan Golshani and Mehrdad Ashtiani. 2021. Proactive auto-scaling for cloud environments using temporal convolutional neural networks. *J. Parallel and Distrib. Comput.* 154 (2021), 119–141.

[12] Xiao He, Ye Li, Jian Tan, Bin Wu, and Feifei Li. 2023. OneShotSTL: One-Shot Seasonal-Trend Decomposition For Online Time Series Anomaly Detection And Forecasting. *Proc. VLDB Endow.* 16, 6 (2023), 1399–1412. https://doi.org/10.14778/3583140.3583155

[13] Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20, 1 (2004), 5–10.

[14] Maowei Jiang, Pengyu Zeng, Kai Wang, Huan Liu, Wenbo Chen, and Haoran Liu. 2023. FECAM: Frequency enhanced channel attention mechanism for time series forecasting. *Advanced Engineering Informatics* 58 (2023), 102158.

[15] Md Ebtidaul Karim, Mirza Mohd Shahriar Maswood, Sunanda Das, and Abdullah G Alharbi. 2021. BHyPreC: a novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine. *IEEE Access* 9 (2021), 131476–131495.

[16] Tahseen Khan, Wenhong Tian, Shashikant Ilager, and Rajkumar Buyya. 2022. Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. *Future Generation Computer Systems* 128 (2022), 320–332.

[17] Reihaneh Khorsand, Mostafa Ghobaei-Arani, and Mohammadreza Ramezanpour. 2018. WITHDRAWN: a fuzzy auto-scaling approach using workload prediction for MMOG application in a cloud environment.

[18] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[21] Anoop S Kumar and Somnath Mazumdar. 2016. Forecasting HPC workload using ARMA models and SSA. In *2016 International conference on information technology (ICIT)*. IEEE, 294–297.

[22] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[23] Patricio Lamas and Erik Demeulemeester. 2016. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling* 19 (2016), 409–428.

[24] Habte Lejebo Leka, Zhang Fengli, Ayantu Tesfaye Kenea, Abebe Tamrat Tegene, Peter Atandoh, and Negalign Wake Hundera. 2021. A hybrid cnn-lstm model for virtual machine workload forecasting in cloud data center. In *IEEE International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 474–478.

[25] Shengming Li, Ying Wang, Xuesong Qiu, Deyuan Wang, and Lijun Wang. 2013. A workload prediction-based multi-vm provisioning mechanism in cloud computing. In *Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 1–6.

[26] Yuxuan Liang, Yutong Xia, Songyu Ke, Yiwei Wang, Qingsong Wen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. 2023. Airformer: Predicting nationwide air quality in china with transformers. In *AAAI conference on artificial intelligence*, Vol. 37. 14329–14337.

[27] Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. 2024. Sparsetsf: Modeling long-term time series forecasting with 1k parameters. *arXiv preprint arXiv:2405.00946* (2024).

[28] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems* 35 (2022), 5816–5828.

[29] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2022. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

[30] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625* (2023).

[31] Federico Lombardi, Andrea Muti, Leonardo Aniello, Roberto Baldoni, Silvia Bonomi, and Leonardo Querzoni. 2019. Pascal: An architecture for proactive auto-scaling of distributed services. *Future Generation Computer Systems* 98 (2019), 342–361.

[32] Laura R Moore, Kathryn Bean, and Tariq Ellahi. 2013. Transforming reactive auto-scaling into proactive auto-scaling. In *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*. 7–12.

[33] Hoang Minh Nguyen, Sungpil Woo, Janggwan Im, Taejoon Jun, and Daeyoung Kim. 2016. A workload prediction approach using models stacking based on recurrent neural network and autoencoder. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 929–936.

[34] Zelin Ni, Hang Yu, Shizhan Liu, Jianguo Li, and Weiyao Lin. 2023. Basisformer: Attention-based time series forecasting with learnable and interpretable basis. *Advances in Neural Information Processing Systems* 36 (2023), 71222–71241.

[35] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

[36] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[38] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. 2024. Fredformer: Frequency debiased transformer for time series forecasting. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2400–2410.

[39] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *Proc. VLDB Endow.* 17, 9 (2024), 2363–2377.

[40] Bane Raman Raghunath and B Annappa. 2015. Virtual machine migration triggering using application workload prediction. *Procedia Computer Science* 54 (2015), 167–176.

[41] Dymitr Ruta, Ling Cen, and Quang Hieu Vu. 2020. Deep bi-directional LSTM networks for device workload forecasting. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 115–118.

[42] Deepika Saxena and Ashutosh Kumar Singh. 2021. Workload forecasting and resource management models based on machine learning for cloud computing environments. *arXiv preprint arXiv:2106.15112* (2021).

[43] RS Shariffdeen, DTSP Munasinghe, HS Bhathiya, UKJU Bandara, and HMN Dilum Bandara. 2016. Adaptive workload prediction for proactive auto scaling in PaaS systems. In *IEEE International Conference on Cloud Computing Technologies and Applications (CloudTech)*. 22–29.

[44] Binbin Song, Yao Yu, Yu Zhou, Ziqiang Wang, and Sidan Du. 2018. Host load prediction with long short-term memory in cloud computing. *The Journal of Supercomputing* 74 (2018), 6554–6568.

[45] Fan-Keng Sun and Duane S Boning. 2022. FreDo: Frequency Domain-based Long-Term Time Series Forecasting. *arXiv e-prints* (2022), arXiv–2205.

[46] D. Sundararajan. 2025. *The Discrete Fourier Transform*. Springer Nature Singapore, Singapore, 37–67. https://doi.org/10.1007/978-981-96-1078-5_2

[47] Gökalp Urul. 2018. *Energy efficient dynamic virtual machine allocation with CPU usage prediction in cloud datacenters*. Master's thesis. Bilkent Universitesi

(Turkey).

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[49] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

[50] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381* (2022).

[51] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations*.

[52] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[53] Zhijian Xu, Ailing Zeng, and Qiang Xu. 2024. FITS: Modeling Time Series with $10k$ Parameters. arXiv:2307.03756 [cs.LG]

[54] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, and Junliang Chen. 2013. Workload predicting-based automatic scaling in service clouds. In *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, 810–815.

[55] Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. 2024. Filternet: Harnessing frequency filters for time series forecasting. *Advances in Neural Information Processing Systems (NeurIPS)* 37 (2024), 55115–55140.

[56] Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. 2023. Frequency-domain MLPs are More Effective Learners in Time Series Forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[57] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[58] Yunhao Zhang and Junchi Yan. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

[59] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

[60] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. 2022. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems* 35 (2022), 12677–12690.

[61] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. PMLR, 27268–27286.