A Self-decoupled Interpretable Prediction Framework for Highly-Variable Cloud Workloads

Bingchao Wang[1,2,3], Xiaoyu Shi[1,2(✉)], and Mingsheng Shang[1,2]

[1] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
xiaoyushi@cigit.ac.cn

[2] Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China

[3] Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Abstract. Cloud workloads prediction plays a crucial role in the various tasks of cloud computing, such as resource scheduling, performance optimization, cost management, etc. However, current time series prediction methods suffer instability and inefficiency issues when addressing cloud workloads, due to the high variability of workload patterns and the high fluctuation within a workload. To address these issues, we propose DeIP4CW, a Self-Decoupled Interpretable Prediction framework for highly-variable Cloud Workloads. It can accurately forecast future job arrival rates in a cloud environment. The core idea of DeIP4CW is to first introduce the periodic and residual states as hidden variables to decouple complicated dependencies in cloud workload signals. Then it adopts a deep expansion learning framework with the block structure to perform workload prediction layer by layer. Each block consists of some *periodic* modules and some *compensation* modules. The *periodic* module with a self-attention mechanism can effectively capture the global trend of cloud workload, while the *compensation* module is employed to compensate for the local volatility information. Moreover, our two customized modules also have interpretable abilities, such as attributing the predictions to either global trends or local compensation. We conduct extensive experiments on the real-world cloud workload traces to evaluate the effectiveness of the proposed DeIP4CW. The experimental results demonstrate the DeIP4CW achieves significant improvements over the best baseline in most cases, and the error reduction can even reach up to 20.66%.

Keywords: Cloud computing · Workload prediction · Time series forecasting · Deep learning

1 Introduction

Cloud computing has become one of the most prevailing computing paradigms in IT society [1]. With the unique characteristic of virtualization technologies,

X. Wang et al. (Eds.): DASFAA 2023, LNCS 13943, pp. 588–603, 2023.
https://doi.org/10.1007/978-3-031-30637-2_39

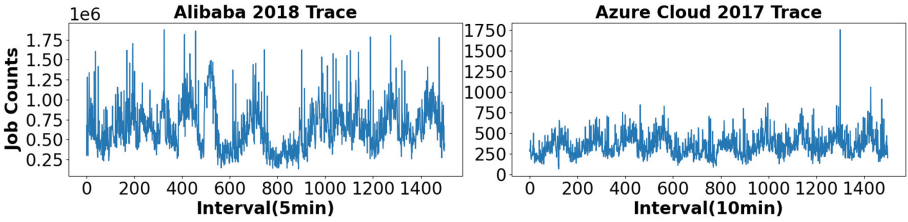

Fig. 1. Cloud workload traces. (a) Alibaba-Cluster-trace-v2018, (b) Azure cloud 2A017 trace.

cloud computing promises an on-demand provisioning mode to satisfy the variety of service level agreements (SLAs) between cloud service providers (CSPs) and end users. In the cloud environment, cloud workloads are usually unstable and not fixed in terms of submitted job rate and arriving time. The variations of cloud workload lead to the over/under-provisioning of resources, which causes unnecessary resource wasting or poor SLAs [2,3]. Therefore, it requires the CSPs to rapidly adjust the resource provisioning solutions for meeting SLAs while saving usage costs.

To achieve these objectives, accurately predicting future job arrival rate is crucial for the performance and cost optimization task in cloud computing. Generally, CSPs can estimate the amount of resources to be allocated in advance through the workload prediction method, so as to prevent waste and poor SLAs caused by over/under-provisioning resources. However, the high variability and fluctuation of cloud workloads make it difficult to accurately predict future job arrival rates:

- **High variability of workload patterns.** Workload patterns in cloud environments usually vary on different cloud-based applications. Figure 1 shows the workload traces collected from Alibaba and Microsoft Azure cloud data centers. Obviously, Alibaba's workload pattern exhibits random changes in terms of job arrival time and rate. Compared to Ali's cluster data, cloud workloads from Azure show a mixture of characteristics with high fluctuation and seasonality [4].
- **High fluctuation within a cloud workload.** Cloud workloads usually show a high dynamic and unexpected burst regarding the submitted job rates and arriving time. As shown in Fig. 1, it demonstrates that the workloads of the cluster fluctuate significantly at different times of the day. Especially cloud workloads will increase dramatically in a short time when an important live show or emergency event happened.

To address the aforementioned challenges, the complicated dependencies hidden in cloud workloads signals should be effectively captured. Based on the learned dependencies between adjacent signals, several methods have been proposed to design cloud workload predictors. Most of the existing methods focus on probabilistic or statistical theories to build their workload predictor,

which only considers the linear dependence on historical observations [5,6]. With their strong ability in sequence data analysis, recurrent neural networks (RNN) have been employed to capture the complicated dependencies hidden in cloud workloads [7–9]. However, these RNN-based methods show poor performance on workloads with unexpected bursts, because of the strong assumption on the dependence between adjacent observations. Thus they cannot make an accurate prediction on highly-variable cloud workloads. In this paper, we propose DeIP4CW, a self-decoupled interpretable prediction framework for highly-variable cloud workloads. We argue that the complicated dependencies of cloud workload signals consist of the inherent periodicity and the correlation between adjacent historical observations. Therefore, the core idea of DeIP4CW is to decouple the complicated dependencies into two dedicated parts, by introducing the periodic and residual states as hidden variables. Inspired by residual learning, we design a deep expansion learning framework with a block structure to achieve this goal, which can perform the workload prediction layer by layer. Each block includes two customized modules, named *periodic* module and *compensation* module. The *periodic* module adopts the self-attention network to capture the global trend and periodic signal hidden in workloads. To cope with the residual signal, the *compensation* module then leverage a couple of one-dimensional convolutional neural networks to capture the local features. By adjusting the number of two modules, the complicated dependencies of cloud workloads can be easily handled.

We divide the three datasets from Alibaba and Azure into nine datasets with different time granularities to test the performance of our DeIP4CW. The results show that DeIP4CW reduces the error rate by an average of 9.00% compared to the primary baseline, and reduces the error rate by up to 20.66% on the dataset azure2019_60min. Besides, we also conduct ablation experiments and hyperparameter-sensitive experiments to verify the effectiveness of our proposed method.

We summarize our major contributions as follows.

- We introduce a self-decoupled prediction framework for highly-variable cloud workloads, which is important but neglected by existing approaches. We decompose the complicated dependencies of cloud workloads into a combination of periodic and residual signals.
- We propose DeIP4CW, a deep expansion learning framework with block structure, to perform cloud workload prediction layer by layer. Each block is composed of two customized modules to capture the global trend and local fluctuation signals. Moreover, these customized modules also have interpretable abilities.
- We conduct extensive experiments on real-world cloud workload trace to validate the effectiveness and adaptively ability of DeIP4CW. The results demonstrate the DeIP4CW outperforms SOTA methods when addressing the highly-variable cloud workloads, and we also visualize the prediction results to interpret model behaviors.

The rest of this paper is organized as follows. The related works are presented in Sect. 2, and the problem formulation is given in Sect. 3. We present the DeIP4CW in Sect. 4 and evaluate its performance in Sect. 5. The conclusion is given in Sect. 6.

2 Related Work

Workload prediction has received extensive attention in cloud computing. At an early stage, researchers focus on employing probabilistic and statistical methods to conduct workload predictors, including auto-regressive model (AR) [10], auto-regressive moving average model (ARMA) [11], exponentially weighted moving averages [12], and others [13,14]. These methods can only capture the linear dependence on historical observations and do not fit well with fluctuations in time series data. As a result, most of them show poor SLAs when addressing high-volatility cloud workloads.

With the development of artificial neural networks, researchers try to learn the complicated dependencies of cloud workloads by building complex neural network structures. Gao et al. [5] introduced a clustering based workload prediction method, which first clusters all the tasks into several categories and then train a neural network-based prediction model for each category respectively. Jitendra et al. [15] offered a workload prediction model using a neural network with a self-adaptive differential evolution algorithm. However, this method is mainly designed for the workload of the HTTP server. The workload pattern of the HTTP server has a strong seasonality, while the workload pattern of the cloud server usually is random. Thus, it is difficult to accurately predict the cloud workload. To deal with the could workloads, Shivani et al. [4] presented a novel time-series forecasting model called *WGAN-gp*, which adopts a Transformer network as a generator and a multi-layer perceptron as a critic. However, the computational complexity will increase for long sequence data. Chen et al. [16] proposed *L-PAW* that leverages the gated recurrent unit (GRU) to achieve accurate prediction for cloud workloads. Meanwhile, Jayakumar et al. [7] proposed *LoadDynamics* that employs the LSTM model to build their self-optimization generic prediction framework for cloud workloads, which can automatically optimize its internal parameters for different workload patterns. However, these RNN-based methods often perform poorly on highly-variable cloud workloads, because of the strong assumption on the dependence between adjacent observations.

3 Problem Formulations

Generally, cloud workload is a count of job arrivals sampled by a physical server or virtual machine (VM) at a constant sampling frequency. Therefore, cloud workload prediction is essentially a time series forecasting problem. Let x_i denote the job arrival counts (JAC) on the cloud servers at the i th moment. The classic autoregressive prediction formula is to project the historical observation

$x_{t-L:t} = [x_{t-L}, \ldots, x_{t-1}]$ into the JAC at the next moment x_t [17]. In [4,18], the Markov property is directly followed to predict the JAC at the next moment, and we also follow this assumption. That is, the JAC at the next moment is only dependent on $x_{t-L:t}$ and has nothing to do with $x_{0:t-L}$. Thus, the workload prediction can be defined as:

$$x_t = f_\theta(x_{t-L:t}) + \epsilon_t, \tag{1}$$

where L is the length of the historical observation sequence, $f_\theta : \mathbb{R}^L \to \mathbb{R}^1$ is a mapping function parameterized by θ, and ϵ_t denotes a value of independent and identically distributed Gaussian noise. Existing methods for period time series (PTS) adopt a few different instantiations of f_θ. For example, Jayakumar et al. [7] employs an LSTM that can automatically optimize its internal parameters. Feng et al. [19] adopted a deep learning prediction model designed with a deep belief network (DBN) composed of multiple-layered restricted Boltzmann machines (RBMs) and a regression layer to fit future workload to historical series. However, Calheiros et al. [14] employs the traditional mathematical and statistical method ARIMA.

4 DeIP4CW

In this section, we introduce the proposed DeIP4CW framework in detail. First, we start with a decoupled formulation of Eq. (1) in Sect. 4.1. Then, we illustrate the details of the deep expansion structure in Sect. 4.2. Last, the composition and neural structure of the *periodic* module and the *compensation* module are explained in Sect. 4.3 and Sect. 4.4.

4.1 The Decoupled Formulation

In response to two challenges of predicting cloud workloads, i.e., high variability of workload patterns and high fluctuation within a workload, we introduce a new function to explore the diversified local fluctuation compositions, besides the main prediction function f_θ in Eq. (1). As a result, we decouple the cloud workload prediction into two independent parts (i.e., f_θ and g_φ), by redefining (1). f_θ is used to capture the variant periodic patterns of cloud workloads, and g_φ is designed to compensate for diversified local fluctuation compositions. Thus, the cloud workload prediction can be formatted as:

$$x_t = f_\theta(x_{t-L:t}) + g_\varphi(x_{t-L:t} - f_\theta(x_{t-L:t})) + \epsilon_t, \tag{2}$$

where $g_\varphi : \mathbb{R}^L \to \mathbb{R}^1$ is a mapping function parameterized by φ that produces a prediction from the difference between the prediction of f_θ and the true input.

4.2 The Deep Expansion Structure

To capture the complicated dependencies (i.e., variant periodic patterns and diversified local fluctuation compositions) of cloud workloads, the key is to trade

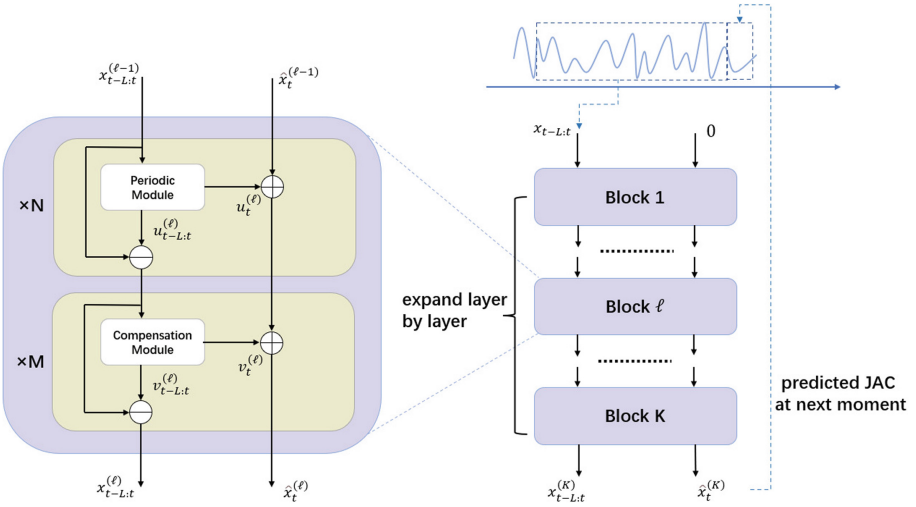

Fig. 2. The whole figure describes the deep expansion structure, and the left part describes the detailed data flow.

off between model capacity and generalization. To achieve this target, deep expansion learning has attracted lots of attention for time-series prediction. N-Beats [20] first proposed a deep expansion structure in time series forecasting, which is a deep neural architecture based on bi-directional residual links. It has demonstrated superior performances on time series forecasting than the traditional methods. Based on N-Beats, DEPTS [17] further introduced the periodic state to decouple the complicated dependencies of time series workloads for a more efficient and accurate prediction. Inspired by these successful examples and combined with the characteristics of cloud workload, we develop a novel framework DeIP4CW based on deep expansion learning for cloud workload prediction. DeIP4CW is also a deep neural network architecture with a block structure. Based on the redefined decoupled formulation (2), each block contains two kinds of customized and parameterized modules: *periodic* module f_θ and *compensation* module g_φ. Figure 2 depicts the architecture of DeIP4CW. Among them, the number of two kinds of modules in each block is controlled by the corresponding hyperparameters N and M. Each block has two residual branches, where $x_{t-L:t}^{(\ell)}$ is the residual after the decomposition of the previous ℓ layers, and $\hat{x}_t^{(\ell)}$ is the accumulation of the prediction results of the previous ℓ layers, $\ell \in [1, K]$.

First, we introduce the update equation of $x_{t-L:t}^{(\ell)}$, whose purpose is to exclude the information that the previous ℓ layers are used for prediction. To be more concrete, $f_\theta^{(\ell j)}$ takes $x_{t-L:t}^{(\ell-1)} - \sum_{q=0}^{j-1} u_{t-L:t}^{(\ell q)}$ as input, where $f_\theta^{(\ell j)}$ denotes the j th *periodic* module in the ℓ th block. The output of *periodic* module can be divided into two parts: $u_{t-L:t}^{(\ell j)}$ and $u_t^{(\ell j)}$, $u_t^{(\ell j)}$ is the prediction based on the current input information, and the other part $u_{t-L:t}^{(\ell j)}$ is called the backcast,

which represents the information used to generate the current prediction $u_t^{(\ell j)}$. $u_{t-L:t}^{(\ell q)} = 0$ when $j = 1$, $j \in [1, N]$. For the *compensation* module, its purpose is to continue to extract information from the residual after N *periodic* modules processing. The input of $g_\varphi^{(\ell k)}$ is $x_{t-L:t}^{(\ell-1)} - \sum_{j=0}^{N} u_{t-L:t}^{(\ell j)} - \sum_{p=0}^{k-1} v_{t-L:t}^{(\ell p)}$, where $g_\varphi^{(\ell k)}$ denotes k th *compensation* module in the ℓ th block, $k \in [1, M]$. The output of *compensation* module is also divided into two parts: $v_{t-L:t}^{(\ell k)}$, $v_t^{(\ell k)}$, where represents prediction result $v_t^{(\ell k)}$ at time t and the prediction result $v_{t-L:t}^{(\ell k)}$ by the residual information, respectively. After that, we update $x_{t-L:t}^{(\ell)}$ by further subtracting $v_{t-L:t}^{(\ell k)}$ from $x_{t-L:t}^{(\ell-1)} - \sum_{j=0}^{N} u_{t-L:t}^{(\ell j)} - \sum_{p=0}^{k-1} v_{t-L:t}^{(\ell p)}$ as $x_{t-L:t}^{(\ell)} = x_{t-L:t}^{(\ell-1)} - \sum_{j=0}^{N} u_{t-L:t}^{(\ell j)} - \sum_{k=0}^{M} v_{t-L:t}^{(\ell k)}$. In addition, we obtain the prediction results of this block by adding the output of the *periodic* module $\sum_{j=0}^{N} u_t^{(\ell j)}$ and the compensation information proposed through the residuals $\sum_{k=0}^{M} v_t^{(\ell k)}$ as $\hat{x}_t^{(\ell)} = \sum_{j=0}^{N} u_t^{(\ell j)} + \sum_{k=0}^{M} v_t^{(\ell k)}$.

Note that the input of the ℓ th block is the output of the $\ell-1$ th block $x_{t-L:t}^{(\ell-1)}$, where the input of the first block is the real historical observation $x_{t-L:t}$, and the output of the last block is the final prediction result $\hat{x}_t$. According to the above description, the structure in Fig. 2 can be summarized as the Eq. (3):

$$\begin{aligned} x_{t-L:t} = x_{t-L:t}^{(0)} &= x_{t-L:t}^{(K)} + \sum_{i=1}^{K}\Big(\sum_{j=1}^{N} u_{t-L:t}^{(ij)} + \sum_{k=1}^{M} v_{t-L:t}^{(ik)}\Big), \\ \hat{x}_t = \hat{x}_t^{(K)} &= \sum_{i=1}^{K}\Big(\sum_{j=1}^{N} u_t^{(ij)} + \sum_{k=1}^{M} v_t^{(ik)}\Big), \end{aligned} \tag{3}$$

where $x_{t-L:t}^{(K)}$ is considered irrelevant information for prediction and will be discarded after the last layer.

Connections and Differences to N-Beats. N-Beats directly uses the historical sequence $x_{t-L:t}$ to predict the value x_t at the next moment. Unlike them, we decouple the historical sequence into two states, period and residual to predict respectively. This brings about changes in the network structure. Compared with N-Beats, which only contains a fully connected network in each block, we introduce a self-attention mechanism and a one-dimensional convolutional neural network to model the periodicity and residual respectively.

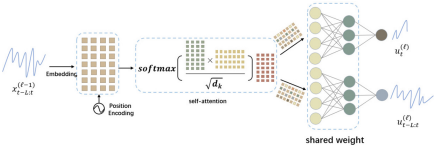

Fig. 3. The *periodic* module

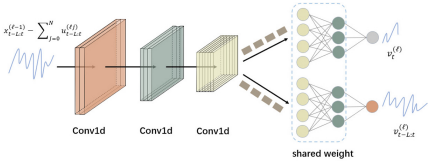

Fig. 4. The *compensation* module.

4.3 The Periodic Module

To model the complicated dependences of workload sequence on periods, we model variant periodic patterns in the sequence through a parameterized function f_θ. With the success of Transformer in many fields, the self-attention mechanism has demonstrated a strong ability in capturing the long-term dependencies [21,22]. Thus, we employ the self-attention mechanism to construct our periodic module.

The structure of the *periodic* module is shown in Fig. 3. Taking ℓ th block, for example, adopts the learned workload feature $x_{t-L:t}^{(\ell-1)}$ after previous $\ell-1$ blocks as input, and output two parts $u_t^{(\ell)}$ and $u_{t-L:t}^{(\ell)}$. In detail, it first encode $x_{t-L:t}^{(\ell-1)}$ with the position information as $x_{t-L:t}^{(\ell-1)'}$. Then $x_{t-L:t}^{(\ell-1)'}$ is fed into the self-attention network to calculate the correlation between each value of the whole sequence. After that, it adopts two standard fully connected networks with ReLU activation to calculate the final prediction result $u_t^{(\ell)}$ and $u_{t-L:t}^{(\ell)}$ respectively. Two neural network shares the same parameters and uses linear projection functions to fit periodic functions and prevailing trends contained in series.

4.4 The Compensation Module

To solve the problem of inaccurate prediction caused by mutation of highly-variable workload, we propose another parameterized function g_φ to extract the remaining information after *periodic* module prediction to compensate for local prediction results. In other methods, the result of the *periodic* module is used directly as the final result [4,10]. However, we found that the data used to predict the final outcome was not fully extracted, especially the highly-variable workload. Therefore, we decouple it into periodic and residual states and propose a *compensation* module to compensate the local prediction result for the global prediction result.

So we introduce a 1D convolutional neural network (Conv1d), which is responsible for extracting local features to compensate for information not noticed during global prediction. Due to the highly-variable of cloud workload, the load values at each moment are not highly correlated, and Conv1d is very effective in this regard.

As with the input of the *periodic* module, we only discuss the case of $k=1$, so the input of the *compensation* module is $x_{t-L:t}^{(\ell-1)} - \sum_{j=0}^{N} u_{t-L:t}^{(\ell j)}$. The specific demonstration is shown in Fig. 4. First, using a three layers Conv1d, we can obtain local features of residual information that cannot be extracted by multiple *periodic* modules. Then, two fully connected networks with ReLU activation with shared parameters and two linear mapping functions are used to obtain forecast and backcast respectively:

$$
\begin{aligned}
feature^{1}_{feature\_size^1} &= AvgPool1d(ReLU(Conv1d^1(x^{\ell-1}_{t-L:t} - \sum_{j=0}^{N} u^{(\ell j)}_{t-L:t}))),\\
feature^{2}_{feature\_size^2} &= AvgPool1d(ReLU(Conv1d^2(feature^{1}_{feature\_size^1}))),\\
feature^{3}_{feature\_size^3} &= AvgPool1d(ReLU(Conv1d^3(feature^{2}_{feature\_size^2}))),\\
feature^{4}_{feature\_size^4} &= FC(feature^{3}_{feature\_size^3}),\\
v^{(\ell)}_{t-L:t} &= LINEAR(feature^{4}_{feature\_size^4}),\\
v^{(\ell)}_{t} &= LINEAR(feature^{4}_{feature\_size^4}),
\end{aligned}
\tag{4}
$$

5 Experiments

Our experiments aim to address two questions: 1). How much benefit does DeIP4CW gain from predicting on highly-variable workload compared to the optimal algorithm? 2). What interpretability can be brought through our two custom modules? To answer the first question, we conduct extensive experiments on real datasets in Sect. 5.2. Then we experiment with the second question in Sect. 5.3 and analyze it for specific cases.

5.1 Experiments Setup

Workload Datasets. We collected three cloud workloads with different fluctuation datasets from two CSPs to evaluate the performance of the algorithm. As shown in Fig. 5, the workload value of alibaba 2018[1] is extremely large, the periodicity is not obvious, and there will be huge fluctuations at a certain moment. Cloud VM workloads from Azure[2] show a mixture of characteristics having high fluctuations and seasonality [4].

The different granularity of cloud workloads will also have a greater impact on workload characteristics. Therefore, we divided the selected three datasets into different time granularities, resulting in 9 different workloads. Among them, we use 1, 5, and 10 min intervals for alibaba 2018, use 10, 30, and 60 min for azure 2017, and 1, 30, and 60 min for azure 2019.

Hyperparameters. We use *Pytorch* to implement our algorithm. When training the algorithm, we use Adam optimizer on NVIDIA GeForce RTX 1080 machine, learning rate = 0.01, loss = L1Loss, epochs = 8000. We use grid search to find the optimal hyperparameters for training DeIP4CW, and their search ranges are shown in Table 1. For other baseline algorithms, we expand the search range based on the default parameters to achieve the best prediction

[1] https://github.com/alibaba/clusterdata.
[2] https://github.com/Azure/AzurePublicDataset.

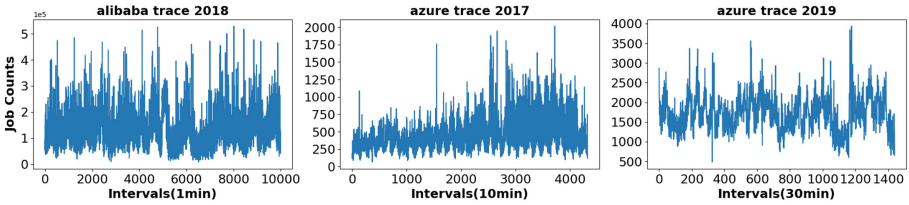

Fig. 5. Cloud Workload Traces. (a) Alibaba Cluster Workload Traces (b) Azure 2017 VM Traces (c) Azure 2019 VM Traces

Table 1. Hyperparameter search space of DeIP4CW algorithm.

Workload	Prediction module size(N)	Residual-aware module size(M)	Block size(K)	Lookback size	Batch size	Kernel size
alibaba 2018	[1, 2]	[1, 2]	[2-30]	[10-48]	[16-1024]	[2-32]
azure 2017				[16-72]	[32, 64]	[2-18]
azure 2019					[32, 64, 128]	

effect. For our network structure, there are several key hyperparameters that have a great influence on the final prediction results, and we conduct a series of experiments on them in Sect. 5.2.

Evaluation Metric. We use Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) to compare the prediction accuracy of different algorithms on all datasets. These two metrics are usually used to measure the similarity between the predicted results and the actual values [4,17].

$$MAPE = 100 \times \frac{1}{n}\sum_{i=1}^{n}|\frac{\widetilde{y}_i - y_i}{y_i}|, RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widetilde{y}_i - y_i)^2}, \tag{5}$$

where n is the total number of data points, $\widetilde{y}_i$ represents predicted JAC at time step i, and y_i represents actual JAC at time step i.

Baselines. We adopt the SOTA deep learning time series forecasting architecture N-Beats [20] as the primary baseline. Furthermore, we use WGAN-gp Transformer [4] for highly-variable cloud workload prediction, which is one of the SOTA cloud workload prediction methods. Of course, we also take into account traditional algorithms such as RNN [23], LSTM [8], GRU [24], ARIMA [6], etc., and fully compare their positions in the field of cloud workload prediction.

Besides, to understand the critical designs of our proposed method, we conduct ablation experiments in Sect. 5.2. We employ three variants of DeIP4CW:

- **DeIP4CW-OP:** The *compensation* module is removed from the entire structure so that the algorithm only retains the *periodic* module.
- **DeIP4CW-OR:** Similarly, the *periodic* module is removed from the entire structure, so that the algorithm only retains the *compensation* module.

Table 2. MAPE and RMSE of DeIP4CW and other baseline algorithms on different datasets. "%Improve." represents the improvement of DeIP4CW over the best-performing of all baseline algorithms. We highlight the best results in bold and sub-optimal results with underlining.

Workload	alibaba 2018						azure 2017						azure 2019					
Interval	1min		5min		10min		10min		30min		60min		1min		30min		60min	
	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
DeIP4CW	**26.3**	**471.77**	**18.68**	**201.08**	**18.2**	**334.66**	**33.58**	**240.07**	**23.07**	**438.69**	**14.32**	596.71	**17.86**	**72.08**	**14.57**	**310.08**	**11.71**	**505.70**
N-Beats	27.18	509.76	20.03	228.15	18.64	377.86	36.13	254.76	23.59	448.06	17.26	698.58	18.58	76.71	17.72	435.29	14.76	723.33
WGAN	36.9	569.6	31.84	221.05	31.18	444.81	36.72	302.48	28.08	517.08	17.66	591.55	33.36	120.04	19.30	337.39	15.21	629.09
LSTM	27.01	482.68	20.31	221.86	18.31	357.09	36.53	278.12	24.62	470.75	15.25	598.44	18.40	76.63	15.67	323.24	12.55	519.95
RNN	27.61	514.86	20.75	240.46	18.83	355.42	37.27	311.68	24.92	489.09	14.55	**570.83**	19.32	78.23	16.68	338.92	12.9	560.28
GRU	26.5	490.48	20.01	204.88	18.78	385.10	36.96	278.68	23.08	459.39	15.59	573.05	18.34	79.74	15.27	315.17	12.18	536.25
ARIMA	30.3	493.03	25.2	226.63	24.43	417.91	47.23	254.30	26.68	459.50	20.14	718.70	20.7	75.56	21.45	463.24	18.19	792.65
%Improve	↓ 0.75%	↓ 3.81%	↓ 6.64%	↓ 1.85%	↓ 0.60%	↓ 5.90%	↓ 7.05%	↓ 5.59%	–	↓ 2.09%	↓ 1.58%	↑ 4.53%	↓ 2.61%	↓ 4.60%	↓ 4.58%	↓ 1.64%	↓ 3.85%	↓ 2.74%

Table 3. Performance comparisons of DeIP4CW, DeIP4CW-OP, DeIP4CW-OR, and DeIP4CW-SW. We highlight the best results in bold.

Workload	DeIP4CW-OP		DeIP4CW-OR		DeIP4CW-SW		**DeIP4CW**	
	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
alibaba_1min	27.68	547.24	26.98	538.86	27.28	558.51	**26.3**	**471.77**
alibaba_5min	26.79	259.32	19.84	203.92	25.13	232.15	**18.68**	**201.08**
alibaba_10min	22.10	426.94	18.43	342.95	22.19	408.96	**18.20**	**334.66**
azure2017_10min	39.37	265.42	35.85	262.64	37.50	278.14	**33.58**	**240.07**
azure2017_30min	28.69	559.85	23.56	447.20	24.85	508.34	**23.07**	**438.69**
azure2017_60min	22.90	972.31	**14.07**	619.80	20.31	891.40	14.32	**596.71**
azure2019_1min	20.34	88.98	18.25	72.40	19.76	90.49	**17.86**	**72.08**
azure2019_30min	20.75	438.01	15.54	330.78	17.9	371.21	**14.57**	**310.08**
azure2019_60min	18.45	756.72	12.25	523.82	17.75	704.87	**11.71**	**505.70**

– **DeIP4CW-SW:** Swap the positions of the *periodic* module and the *compensation* module, so that the data is processed locally before global prediction.

5.2 Evaluation Results

Comparing DeIP4CW with Competitors. Table 2 shows the prediction errors of DeIP4CW and other baseline algorithms on different datasets. We can find that our algorithm outperforms other algorithms in almost all cases. In particular, on the azure2019_60min dataset, DeIP4CW achieves a 20.66% reduction in prediction error compared to the primary baseline. In order to show the final prediction effect more intuitively, Fig. 6 draws the predicted data and real data of alibaba_1min and azure2017_60min dataset. For a highly-variable dataset like alibaba 2018, the prediction errors of all algorithms are large, but our algorithm can still achieve the best results, due to the cooperation between the *periodic* module and the *compensation* module. We can observe that for the azure2019 dataset, which exhibits very good periodicity, DeIP4CW can also achieve an accurate result.

We note that while our algorithm's MAPE outperforms other baselines on most datasets, RMSE is a bit weaker on individual datasets. Because the RMSE

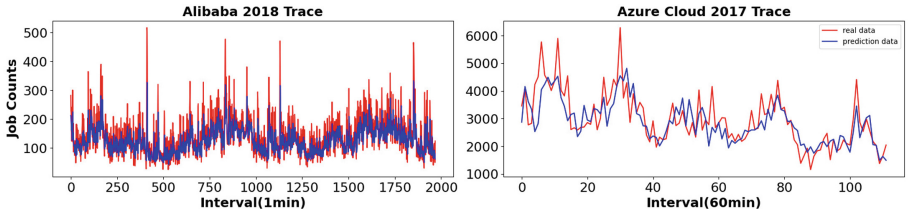

Fig. 6. The predicted data and real data of alibaba_1min and azure2017_60min.

is more sensitive to larger values, and when there are values in the prediction that are much larger than the true value, the RMSE will be larger. Because N-Beats only uses a fully connected network, it cannot capture complex workload pattern changes, but it still achieves good results compared to traditional mathematical statistics methods. RNN and LSTM have storage units that can store more complex historical sequence information, the prediction effect is better than N-Beats, and even the RMSE of individual datasets exceeds our method. WGAN-gp Transformer is one of the most advanced cloud workload prediction algorithms. During the training process, we found that in many cases, its final prediction result is a straight line close to the median line of data fluctuations. Overall, unlike GRU and LSTM, which only perform well on some datasets, our method can handle data with various workload patterns.

Ablation Tests. Then, let us focus on Table 3. We can see that on the highly-variable datasets of azure2017 and alibaba, DeIP4CW-OP cannot predict the data well from a global perspective by using the *periodic* module alone. However, on the dataset with a strong periodicity of azure2019, the effect can be better. Nonetheless, it still has a large error rate compared to our method, which indicates that only predicting the periodicity from the global loses a lot of local information, proving the importance of our proposed *compensation* module. The effect of DeIP4CW-OR using the *compensation* module alone is the closest to the effect of DeIP4CW, and exceeds the performance of DeIP4CW on azure2017_60min. However, its prediction accuracy still falls short of ours. It shows that for different workload patterns, only using local information to predict the results is inaccurate, which proves the importance of our proposed *periodic* module. Although the performance of DeIP4CW-SW in which the positions of the *periodic* module and the *compensation* module are exchanged is better than that of DeIP4CW-OP, it still has a very wrong, which explains why we put the *periodic* module responsible for global prediction at the beginning and then pass the local features to extract useful information from the residuals. In all of our designs, including the *periodic* module, the *compensation* module and the placement of both play a critical role in ultimately accurately predicting future workloads.

Hyperparameter Analysis. In order to verify the sensitivity of our method to hyperparameters, we selected several datasets and conducted hyperparameters

analysis experiments. The final experimental result is shown in Fig. 7. From Fig. 7(a), we can know that when the size of deep expansion layers is too small, the information related to the prediction result cannot be extracted fully. On the contrary, if the size is too large, there will be content unrelated to the prediction in the extracted information, which will eventually affect the accuracy of the prediction result. It should be noted that the optimal number of expansion layers corresponding to the workload with different characteristics is different. It can be seen from Fig. 7(b) that when the ratio of the number of *periodic* modules to the number of *compensation* modules is 1:1, a very low prediction error rate can be obtained. When the number of two modules is unbalanced, the whole structure does not work well. Figure 7(c) shows the relationship between the number of different CNN layers and the final result. We can clearly see that when the number of CNN layers is too small or too many, the local features it extracts either have insufficient information, or the information contains unrelated noises, so we end up using a 3-layer Conv1d in our neural network.

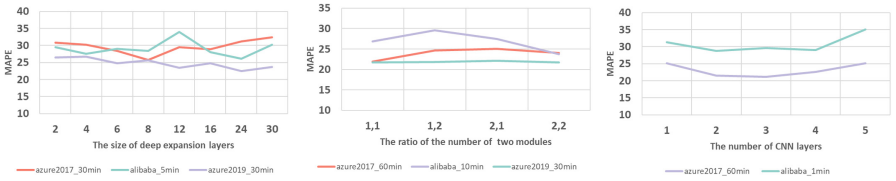

Fig. 7. (a) The effect of different depth expansion layers on different datasets. (b) The prediction results of different combinations of *periodic* modules and *compensation* modules numbers. (c) The prediction performance of different convolutional layers.

5.3 Interpretability

Classical time series forecasting divides the sequence components into seasonal, trend and Gaussian noise [20]. Benefiting from our proposed *periodic* module and *compensation* module, it can bring some interpretability. From the Eq. (3), we attribute the final prediction result to two parts global trends $\sum_{j=1}^{N} u_t^{(\ell j)}$ and local compensation $\sum_{k=1}^{M} v_t^{(\ell k)}$. According to their share of the final result, we can infer the fluctuation characteristics of the current cloud workload. In more detail, if the data has a large fluctuation and the periodicity is not obvious, then the effect of the *periodic* module has a high probability of having a poor effect. And Sect. 5.2 proves that the *compensation* module using Conv1d can extract local features to compensate for the loss that the *prediction* module does not work well.

Figure 8 shows the results we verified in our experiments. For highly-variable aibaba_1min workload, the prediction module works poorly, but thanks to our proposed *compensation* module, they can extract the information missed by the

periodic module. For azure2017_60min workload with obvious periodicity, the *periodic* module can extract most of the data, and the useful information will account for a small proportion of the remaining residuals. This also fully proves that our network structure can reflect the fluctuation characteristics of the data in the final prediction results.

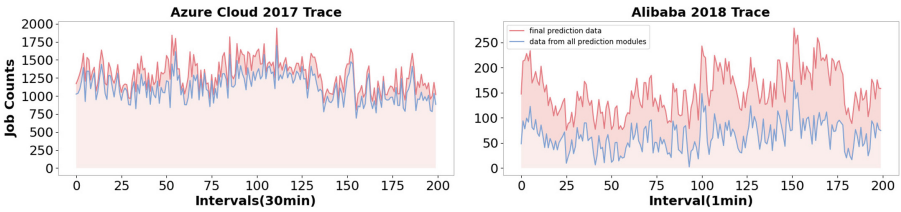

Fig. 8. The orange line in the figure represents the final prediction result, and the blue line represents the sum of the results produced by all prediction modules. The orange shading represents the sum of the information extracted by all *compensation* modules. (Color figure online)

6 Conclusion

In this paper, we propose a self-decoupled interpretable method for predicting highly-variable cloud workloads. Our core contribution is to introduce periodic and residual states to address the incomplete extraction of effective information caused by highly-variable workload predictions and propose a *compensation* module to extract it specifically. Interestingly, the two modules we propose also bring some interpretability. Finally, extensive experiments are conducted on cloud workloads to demonstrate the effectiveness of our method in handling highly-variable data.

Acknowledgments. This work is supported in part by the National Natural Science Foundation of China under Grants 62072429, in part by the Chinese Academy of Sciences "Light of West China" Program, and in part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008, HZ2021017), and the "Fertilizer Robot" project of Chongqing Committee on Agriculture and Rural Affairs.

References

1. Luo, C., et al.: Correlation-aware heuristic search for intelligent virtual machine provisioning in cloud systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 12363–12372 (2021)
2. Ran, L., Shi, X., Shang, M.: SLAs-aware online task scheduling based on deep reinforcement learning method in cloud environment. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1518–1525. IEEE (2019)

3. Zhao, Z., Shi, X., Shang, M.: Performance and cost-aware task scheduling via deep reinforcement learning in cloud environment. In: Troya, J., Medjahed, B., Piattini, M., Yao, L., Fernández, P., Ruiz-Cortés, A. (eds.) ICSOC 2022. LNCS, vol. 13740, pp. 600–615. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20984-0_43
4. Arbat, S., Jayakumar, V.K., Lee, J., Wang, W., Kim, I.K.: Wasserstein adversarial transformer for cloud workload prediction. arXiv preprint arXiv:2203.06501 (2022)
5. Gao, J., Wang, H., Shen, H.: Machine learning based workload prediction in cloud computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–9. IEEE (2020)
6. Nelson, B.K.: Time series analysis using autoregressive integrated moving average (ARIMA) models. Acad. Emerg. Med. **5**(7), 739–744 (1998)
7. Jayakumar, V.K., Lee, J., Kim, I.K., Wang, W.: A self-optimized generic workload prediction framework for cloud computing. In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 779–788. IEEE (2020)
8. Graves, A.: Long short-term memory. In: Graves, A. (ed.) Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385, pp. 37–45. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2_4
9. Kumar, J., Goomer, R., Singh, A.K.: Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. Procedia Comput. Sci. **125**, 676–682 (2018)
10. Dinda, P.A., O'Hallaron, D.R.: Host load prediction using linear models. Cluster Comput. **3**(4), 265–280 (2000)
11. Vecchia, A.V.: Maximum likelihood estimation for periodic autoregressive moving average models. Technometrics **27**(4), 375–384 (1985)
12. Winters, P.R.: Forecasting sales by exponentially weighted moving averages. Manag. Sci. **6**(3), 324–342 (1960)
13. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: 2011 IEEE 4th International Conference on Cloud Computing, pp. 500–507. IEEE (2011)
14. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2014)
15. Kumar, J., Singh, A.K.: Workload prediction in cloud using artificial neural network and adaptive differential evolution. Futur. Gener. Comput. Syst. **81**, 41–52 (2018)
16. Chen, Z., Hu, J., Min, G., Zomaya, A.Y., El-Ghazawi, T.: Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans. Parallel Distrib. Syst. **31**(4), 923–934 (2019)
17. Fan, W., et al.: DEPTS: deep expansion learning for periodic time series forecasting. In: International Conference on Learning Representations (2021)
18. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int. J. Forecast. **36**(3), 1181–1191 (2020)
19. Qiu, F., Zhang, B., Guo, J.: A deep learning approach for VM workload prediction in the cloud. In: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 319–324. IEEE (2016)
20. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: N-beats: neural basis expansion analysis for interpretable time series forecasting. In: International Conference on Learning Representations (2019)

21. Wu, H., Xu, J., Wang, J., Long, M.: AutoFormer: decomposition transformers with auto-correlation for long-term series forecasting. In: Advances in Neural Information Processing Systems, vol. 34, pp. 22419–22430 (2021)
22. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115 (2021)
23. Medsker, L.R., Jain, L.C.: Recurrent neural networks. Des. Appl. **5**, 64–67 (2001)
24. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP (2014)