EN-Beats: A Novel Ensemble Learning-based Method for Multiple Resource Predictions in Cloud

Ming Chen[†], Maria Rodriguez Read[†], Patricia Arroba[§] and Rajkumar Buyya[†]

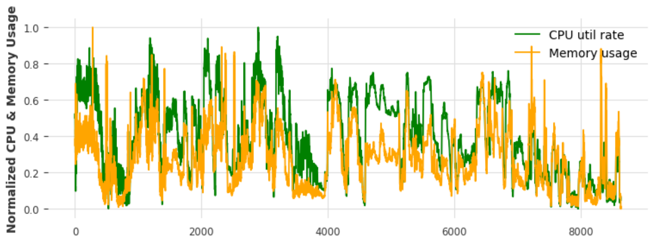
[†]*The Cloud Computing and Distributed Systems (CLOUDS) Laboratory*,
School of Computing and Information Systems, *The University of Melbourne*, Melbourne, Australia
[§]*Laboratorio de Sistemas Integrados (LSI), CCS–Center for Computational Simulation*
Universidad Politécnica de Madrid, Madrid, Spain
[†]mingc4@student.unimelb.edu.au, {maria.read, rbuyya}@unimelb.edu.au, [§]p.arroba@upm.es

***Abstract*—Cloud computing has become an important driving force in the economy and the fundamental facility for digitization transformation. Due to its rapid development and increasing demands, accurate resource usage prediction for the cloud has been a long-term challenge. To address this challenge, this paper introduces *RCorrPolicy* for resource metrics selection and proposes *EN-Beats*, an efficient ensemble learning-based approach, for predicting multiple resource usages in the cloud. The paper presents trace-driven experiments conducted on a real-world dataset, demonstrating notable improvements in predicting multiple resource metrics. Ablation experiments conducted on existing methods for *RCorrPolicy* indicate that the proposed policy enhances the performance of these methods across different evaluated metrics. Furthermore, *EN-Beats* outperforms existing methods by achieving the lowest NRMSE (lower values indicate better performance) for CPU util rate (up to 8% lower), memory usage (up to 6% lower), and network incoming traffic (up to 3% lower). Additionally, *EN-Beats* attains the highest R^2 score (higher values indicate better performance) for predicting CPU util rate (up to 1.39 higher), memory usage (up to 0.57 higher), and network incoming traffic (up to 0.62 higher).**

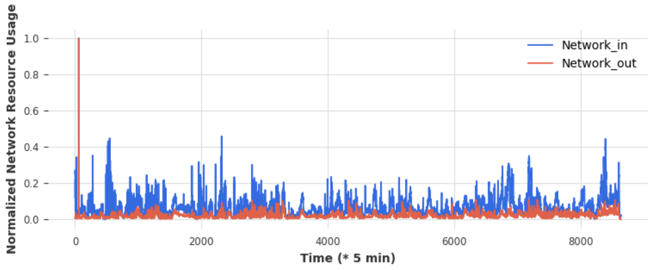
***Index Terms*—resource usage prediction, correlation analysis, ensemble learning**

I. Introduction

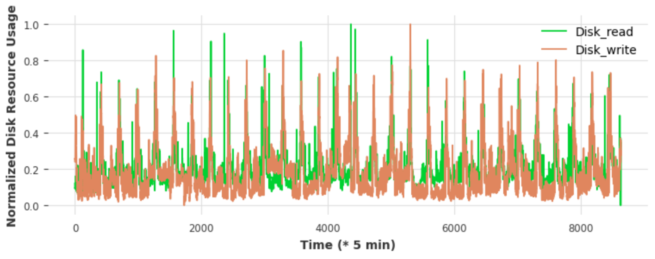
Recently, a variety of devices and cloud-supported applications are being used in every aspect of our life. Smart devices, including surveillance cameras, augmented reality helmets, cell phones and autonomous vehicles, are generating massive data, irrespective of structured data or unstructured data [1]. Similarly, there are also arising complex cloud applications (e.g., WeChat, Instagram, Netflix, and TikTok) producing a huge amount of data. The more we use these applications or services, the smarter of these applications we may feel. It is mainly because the application itself can learn from historical data to automate the decision-making process, like the recommendation features of TikTok. Due to such a trend, numerous new devices and complex applications call for a great surge of vibrant cloud services, thus increasing the consumption of these cloud-supported applications. Moreover, during the global COVID-19 pandemic, many companies and government sectors have accelerated the speed of digitization of their business or services, which also invokes the need for surging cloud resource demands and efficient resource management techniques in cloud systems.



(a) Normalized historical usages of CPU and Memory.



(b) Normalized historical data of network incoming/outgoing traffic.



(c) Normalized historical usages of disk reading and writing.

Fig. 1: An illustration of normalized average resource usages across the 1,250 Virtual Machines (VMs) from a modern cloud at `5 min` interval index in dataset from `Bitbrains`.

To better utilize and manage cloud resources, efficient resource management strategies in cloud data centers are important. We conducted an in-depth experimental analysis

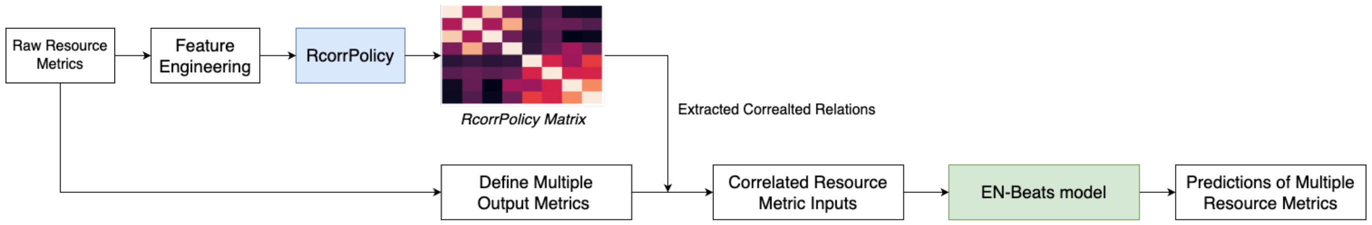

Raw Resource Metrics
Feature Engineering
RcorrPolicy
RcorrPolicy Matrix
Extracted Correalted Relations
Define Multiple Output Metrics
Correlated Resource Metric Inputs
EN-Beats model
Predictions of Multiple Resource Metrics

Fig. 2: The overall workflow of *RCorrPolicy* for selecting resource metrics and *EN-Beats* for resource prediction.

of multiple resource utilization based on open-source `Bitbrains` [2] dataset[1] from a real cloud production environment and we observed that the resource usage fluctuate greatly and demonstrate great uncertainties, as shown in Figure 1. Inefficient resource predictions for fluctuating resource consumption bring out challenges like QoS violations due to resource under-subscription, or inefficient resource utilization when oversubscribing too many resources.

To address the problem of how to efficiently predict resource utilization in dynamic cloud computing environments, different approaches are proposed for different scenarios. However, most of these approaches fail to capture the latent correlations among different resource metrics and have limitations on multiple resource predictions. A few of the latest research works, like [3], do consider resource correlations, but their algorithms for correlation calculation are based on `pearson` correlation which assumes two calculated data vectors are uniformly distributed, while some resources in real clouds might not have these patterns. Moreover, the proposed techniques in [3] fail to make multiple resource usage predictions.

In this paper, we proposed an efficient ensemble learning-based approach *EN-Beats* along with the resource metric selection policy *RCorrPolicy* for multiple resource metrics predictions in cloud computing environments. *RCorrPolicy* is used for selecting the correlated resource metrics, which are adopted as the inputs to *EN-Beats* model for future resource prediction. The ablation experiments have shown the effectiveness of *RCorrPolicy* and the improvement in workload predictions by *EN-Beats* model. Our contributions can be summarized as:

- We proposed a resource metric selection policy *RCorrPolicy* by choosing the correlated metrics from raw data traces.
- We designed the ablation experiments and evaluated the effectiveness of *RCorrPolicy* by presenting ablation experimental results via widely-used current models.
- We proposed the *EN-Beats* model based on ensemble learning by aggregating multiple weak learners into a strong learner for making multiple resource metric predictions.
- We found that the recently proposed *N-Beats* model is a good candidate for cloud resource usage prediction. To the best of our knowledge, *N-Beats* hasn't been analyzed and adopted in the scenario of cloud resource utilization, which we are able to do.

[1]The open-source Bitbrains dataset can be downloaded from http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

The remainder of this paper is organized as follows. Section II discusses the background and motivation of this work. Related work is discussed in the following Section III. In Section IV, we explain feature engineering and the resource correlation selection policy *RCorrPolicy* for choosing the ranked correlated resource metrics. Section V presents the design of *EN-Beats* model by illustrating the model architecture with weak learners and the aggregation process of constructing a strong learner. The evaluation and experimental part are discussed in Section VI, which includes the ablation experiments of *RCorrPolicy* and the performance evaluation of *EN-Beats*. Section VII concludes the paper along with a discussion on future work.

II. Background and Motivation

A. Background

Cloud computing usually refers to the delivery of different resources over the Internet, including resources like memory, CPU, bandwidth, disk, and applications/services. It has emerged as a pivotal driving force in the economy, serving as the foundational infrastructure for digital transformation. Despite the numerous advantages it offers, modern cloud computing also faces a range of challenges that hinder its overall efficiency. One of the primary obstacles lies in accurately predicting workload resource usage in cloud data centers, which is exacerbated by the escalating demands of diverse cloud applications. Consequently, workloads in cloud computing exhibit dynamic and uncertain characteristics, presenting a formidable prediction problem for resource usage. Unfortunately, existing research primarily focuses on single resource metric prediction and fails to capture the underlying correlations among different resource metrics.

B. Motivation

These challenges motivate us to propose a new method for tackling the problem of efficient resource predictions in cloud computing environments. As observed from Figure 3, the resource utilization characteristics within randomly selected Virtual Machines (VMs) are distributed unevenly, where different colors represent different VMs. It is worth noting that while some VMs are highly utilized, others demonstrate a markedly low utilization rate. In addition, the sub-figures in Figure 1 depict that the utilization of correlated resources (e.g.,

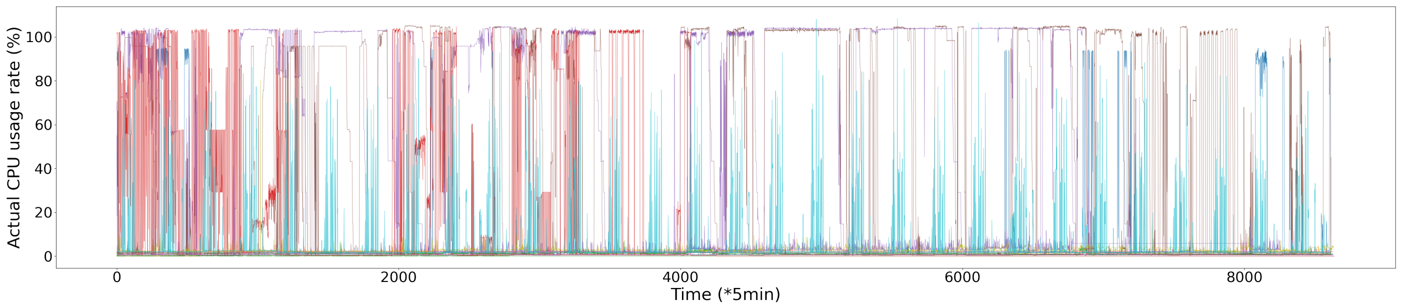


Fig. 3: An illustration of actual CPU utilization rate from 20 randomly selected VMs across 30 days from `Bitbrains` dataset. Different color line represents different CPU utilization rate in the corresponding VM.

CPU and memory, network incoming and outgoing traffic) are showing similar utilization patterns across time, which means that these resource metrics are correlated to some extent. The correlation matrix is shown in Figure 4. These figures from the real cloud production environment demonstrate the imbalance and dynamics in resource allocation and inefficiency in task scheduling.

To alleviate these issues, methods of correlation analysis among different resource metrics and efficient learning techniques should be proposed. Our main motivations can be summarized as follows:

- Most current research works fail to analyze the correlations among different resource metrics and there is no suitable correlation selection policy. Therefore, latent correlation analysis and correlation selection policy should be proposed.
- Some traditional methods like Principal Component Analysis (PCA) and Support Vector Regression (SVR) reduce the feature space largely, resulting in the loss of some valuable features, thus the system model performance is degraded.
- There are limitations to the dominant RNN-based methods (such as LSTM, Bi-LSTM, GRU) for estimating dynamic and long-term resource utilizations due to issues like gradient vanishing and exploding.

III. Related Work

In this section, we summarize the methods employed in previous research on predicting workload resource usage in cloud computing environments. The contributions of these studies can be primarily categorized into three types: regression-based models, learning-based models, and hybrid-based models. Regression-based approaches typically incorporate linear regression and auto-regression models. In terms of learning-based models, machine learning and deep learning-based models are widely utilized across various scenarios. Hybrid-based models generally integrate multiple models to tackle challenges in complex cloud environments.

A. Regression-based Models

In recent years, many researchers have conducted extensive research on resource usage prediction from grid computing to cloud computing. The widely-used regression model for time-series forecasting is ARIMA (Autoregressive integrated moving average) model, which inspired lots of researchers to design ARIMA-based predictive models. Calheiros et al [7] introduced a regression model based on auto-ARIMA for predicting workloads by using real traces from web server requests. Other similar ARIMA-based models addressed this challenge for other scenarios [11] [12] [13]. Bi et al. [14] combined Savitzky-Golay filter and wavelet decomposition for workload prediction in the following time slot. Doulamis et al. [15] proposed a non-linear task prediction by incorporating Quality of Service oriented resource management in grid computing. These regression-based methods showed their effectiveness in terms of resource prediction. Farahnakian et al. [5] introduced a prediction approach by linear regression technique based on the history of resource usage in each host.

B. Learning-based Models

Considering the characteristics of complex workloads, an increasing number of researchers have applied numerous learning-based approaches for addressing this challenge. The learning-based models mainly include machine learning based-models and deep learning-based models.

For machine learning-based models, Gao et al. [6] introduced a clustering-based workload prediction method by clustering all tasks into different categories and training a single prediction model for each category respectively. In [16], authors introduced Bayesian-based model for predicting short and long-term virtual resources by learning workload patterns from several data centers.

Deep learning-based models have been widely used in recent years. Qiu et al. [17] introduced the VM workload prediction model, i.e., Deep Belief Network (DBN), which consists of multiple-layered Restricted Boltzmann Machines (RBMs) and a regression layer. By integrating Extreme Gradient Boosting Tree, Eli et al. [18] introduced a system, known as *Resource Central*, to collect VM characteristics in Azure and learn VM behaviors offline and predict the over-subscription of VM resources. Boris N. Oreshkin et al [8] proposed a novel time-series forecasting model N-BEATS, which solves the problem of predicting univariate time series. LSTM/BiLSTM-based [9] [19] [20] [21], GRU-

TABLE I: A comparison of related work.

Work	**Resource Types**			**Resource Prediction**		**correlation analysis**	**colocated workload**	**Cloud Environments**
	cpu	memory	network	single	multiple			
RPTCN [3]	✓	✗	✗	✓	✗	✓	✗	✓
esDNN [4]	✓	✓	✗	✓	✓	✗	✓	✓
[5]	✓	✗	✗	✓	✗	✗	✗	✓
[6]	✓	✗	✗	✓	✗	✗	✗	✓
ARIMA [7]	✗	✗	✓	✓	✗	✗	✗	✓
N-BEATS [8]	✗	✗	✗	✓	✗	✗	✗	✗
LSTM [9]	✗	✗	✓	✓	✗	✗	✗	✓
TCN [10]	✗	✗	✗	✓	✓	✗	✗	✓
EN-Beats (This work)	✓	✓	✓	✓	✓	✓	✓	✓

based [4] and TCN-based [3] [10] deep learning methods are also widely stuied in academia. Kumar et al [9] adopted LSTM-RNN to predict cloud workload for solving issues of power consumption and resource scaling. Bai et al [10] conducted a systematic experiment evaluation for sequence modeling and found that a simple convolutional architecture outperforms traditional recurrent networks, which inspired the author to design an efficient temporal convolutional neural network for sequence modeling.

C. Hybrid-based Models

To be adaptive to different resource prediction challenges, some researchers attempted to integrate multiple models for tackling sophisticated challenges. In [22], Janardhanan et al. proposed methods for forecasting of CPU usage of machines using LSTM and ARIMA. Ceticik et al. [23] introduced an Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC) by combing statistical and learning methods. Bankole et al. [24] developed prediction models for cloud client prediction in terms of TPC-W benchmark web application through the implementation of Linear Regression (LR), Neural Networks (NN), and Support Vector Machine (SVM). Recently, the Quantum Neural Network-based approach for predicting cloud resource usage has also gained attention. Singh et al. [25] proposed hybrid models by implementing an Evolutionary Quantum Neural Network (EQNN) and integrating a Self Balanced Adaptive Differential Evolution (SB-ADE) model for optimizing qubit network weights, which are implemented for cloud resource prediction as well.

D. Comparisons with EN-Beats

When comparing with the proposed method *EN-Beats*, these models have limitations on correlated latent feature learning and adaptions for multiple resource patterns. Firstly, these models are mainly considering single resource metrics as inputs, thus such models are weak in taking the latent correlations among different resource metrics. Secondly, in the modern cloud, different resources show different time patterns. However, these models mentioned above are usually designed for specific workload pattern learning like periodical web requests modeling and single CPU utilization prediction, while our proposed models show good performance on learning different patterns of resource metrics, like CPU utilization rates, memory usage, and network incoming traffic.

As noted in Table I, compared to related works, our method is unique because it is able to support different usage patterns from multiple types of resources and predictions along with handling correlation analysis and co-located application workloads.

IV. Feature Engineering with RCorrPolicy

In this section, we will first describe the pre-processing of raw cluster traces through normalization. Secondly, we will compare the `Pearson` correlation and `Spearman` correlation. Finally, we will detail our proposed resource metric selection algorithm, referred to as *RCorrPolicy*.

A. Dataset Preprocessing and Feature Scaling

The original `Bitbrains` dataset consists of raw resource provision and utilization metrics from different virtual machines (VMs) in the cloud. As shown in Figure 3, we can observe that, in some VMs, the CPU resource utilization rate remains nearly zero for a significant of time span. Thus, firstly, we pre-processed the raw data by calculating the mean values of the same metrics at the same timestamp among all VMs. Then, we can get the mean resource usage metrics for all VMs across the observed time span. Lastly, to make the model training process more efficient, we normalized the pre-processed data via the min-max method, as indicated in Equation (1)

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

where x is the original value, x_{min} is the minimal value, x_{max} is the maximum value and x_{scaled} is the scaled value. Another nonnegligible step is to scale back the predicted values. After finishing model training, if the predicted scaled values are taken as the resource utilization values, it would be wrong for calculating errors and there are biases for real value predictions, while some researchers may forget to rescale the values especially when predicting CPU or memory utilization rate, which have the same value range (from 0 to 1) as the scaled values. Thus, it is also important to re-scale the predicted scaled value with the following Equation (2).

$$\hat{x} = \hat{x_{scaled}} \cdot (x_{max} - x_{min}) + x_{min} \tag{2}$$

where $\hat{x}$ is the final predicted value, $\hat{x_{scaled}}$ is the predicted scaled value from the trained model.

B. Comparison of Pearson Correlation and Spearman Correlation

We compared the two widely-used statistical correlation methods. The main differences between these two correlation methods lie in the assumptions of the data distribution and the calculated results. `Pearson` correlation assumes that the data are uniformly distributed and the results show linear relations, while there is no requirement for `Spearman` correlation to assume the data to be uniformly distributed and its correlation results show monotonicity between calculated data instead of linearity calculated by `Pearson`. The Equation 3 represents how correlations between two data vectors are calculated by `Spearman` correlation.

$$\rho = 1 - \frac{6\sum_{i\in n}(\boldsymbol{x}-\boldsymbol{y})_i^2}{n(n^2-1)} \tag{3}$$

where ρ represents the correlations between data vector $\boldsymbol{x}$ and vector $\boldsymbol{y}$, i is the ith observation, and n is the total number of observations.

C. Correlation selection policy for resource metrics

Considering the actual cloud computing environments, we design *RCorrPolicy* to analyze the VM metrics correlations and select the metrics for predicting either single or multiple future resource utilization. The policy procedure is described in Algorithm 1.

Algorithm 1 demonstrates the resource metrics correlation selection process from raw data traces. Firstly, from timestamp 1 to t, we collect the raw resource metrics vector $\boldsymbol{R_{res}} = \{\vec{\boldsymbol{R_1}}, \vec{\boldsymbol{R_2}}, \vec{\boldsymbol{R_3}}, ..., \vec{\boldsymbol{R_t}}\}$ from data traces in cloud production environment. Usually, at each timestamp, there are multiple different resource metrics, thus m resource metrics at timestamp t can be formulated as $\vec{\boldsymbol{R_t}} = \{\boldsymbol{r_1^t}, \boldsymbol{r_2^t}, \boldsymbol{r_3^t}, ..., \boldsymbol{r_m^t}\}$. Secondly, based on the observations from the correlation heat map in Figure 4, in our scenario, we define the threshold T, which can be modified to other values under different scenarios. With the defined threshold T, we can select the correlated resource metrics for the target resource metric. Then, by adopting `Spearman` correlation analysis, we calculate the correlations between $\vec{\boldsymbol{r_i}} = \{r_i^1, r_i^2, r_i^3, ..., r_i^t\}$ and $\vec{\boldsymbol{r_j}} = \{r_j^1, r_j^2, r_j^3, ..., r_j^t\}$, which represents the i^{th} and j^{th} resource metric from timestamp 1 to t respectively. Finally, by following the loop process in Algorithm 1, the correlated metric list for i^{th} resource metric will be represented as $List\_Corr_i$ and the final all correlated lists can be represented as $\bigcup_{i=1}^{m} List\_Corr_i$.

V. Design of EN-Beats Model for Resource Prediction

In this section, we will introduce the proposed *EN-Beats* method. According to the observations of traces from `Bitbrains` cloud and intrinsic characteristics of neural networks, we make the following assumptions:

Algorithm 1 *RCorrPolicy* algorithm for resource metrics selection from raw resource metrics matrix

Input: The trace logs record different raw resource metrics in the cloud system from timestamp 1 to t. The raw resource metrics matrix can be represented as $\boldsymbol{R_{res}} = \{\vec{\boldsymbol{R_1}}, \vec{\boldsymbol{R_2}}, \vec{\boldsymbol{R_3}}, ..., \vec{\boldsymbol{R_t}}\}$, where $\vec{\boldsymbol{R_t}} = \{\boldsymbol{r_1^t}, \boldsymbol{r_2^t}, \boldsymbol{r_3^t}, ..., \boldsymbol{r_m^t}\}$ indicates there are m resource metrics at time t.

Output: the selected resource metrics matrix indicates correlated resource metrics.

```
1: Define a threshold $T$ $(0 < T \leq 1)$ for the selecting boundary in initial correlation matrix. $\vec{\boldsymbol{r_i}} = \{r_i^1, r_i^2, r_i^3, ..., r_i^t\}$ represents the $i^{th}$ resource metric from time 1 to $t$. $\vec{\boldsymbol{r_j}} = \{r_j^1, r_j^2, r_j^3, ..., r_j^t\}$ represents the $j^{th}$ resource metric from time 1 to $t$. Define the correlated resource metrics list as $List\_Corr_i$ for $\vec{\boldsymbol{r_i}}$ and initially $List\_Corr_i$ is an empty list.
2: for $i = 1$ to $m$ do
3:     for $j = 1$ to $m$ do
4:         $Corr_{ij} \leftarrow Spearman\ (r_i, r_j)$
5:         if $(\mid Corr_{ij} \mid \geq T)$ then
6:             $List\_Corr_i \leftarrow List\_Corr_i.ADD\{Corr_{ij}\}$
7:         end if
8:     end for
9: end for
10: $\boldsymbol{List\_Corr} \leftarrow \bigcup_{i=1}^{m} List\_Corr_i$
11: return $\boldsymbol{List\_Corr}$
```

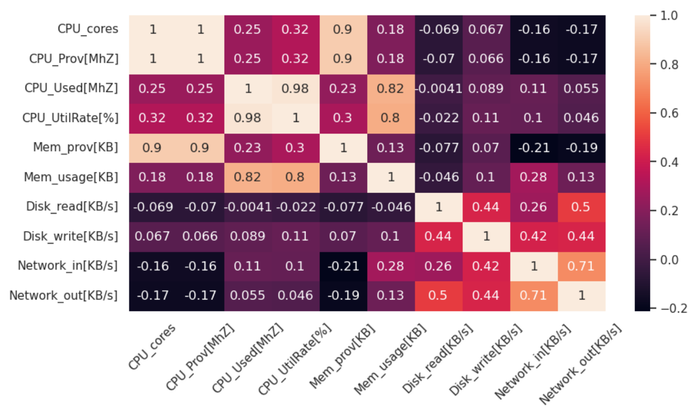


Fig. 4: The `Spearman` correlation heatmap for different resource metrics in our experimental scenario. The metrics targeted for predictions are CPU_UtilRate(%), Mem_usage(KB), and Network_in(KB/s).

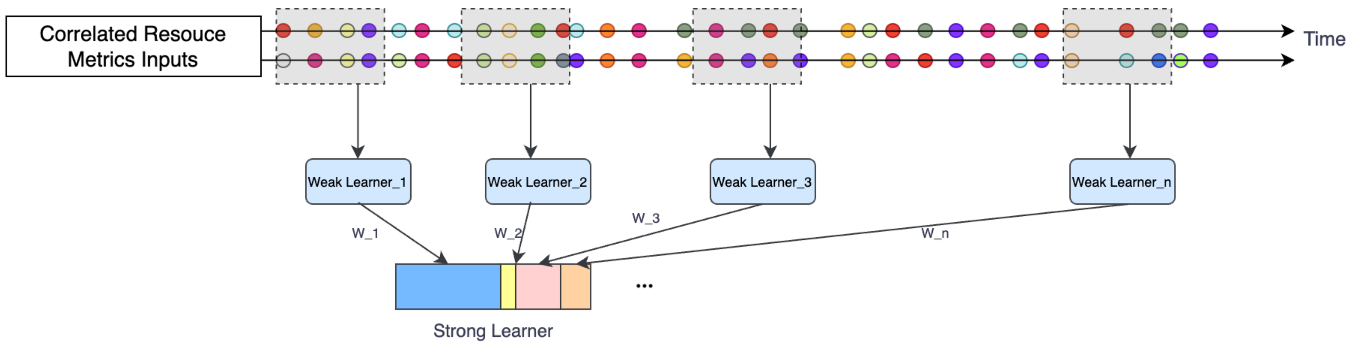


Fig. 5: The overall design framework of *EN-Beats* with strong learner and weak learners.

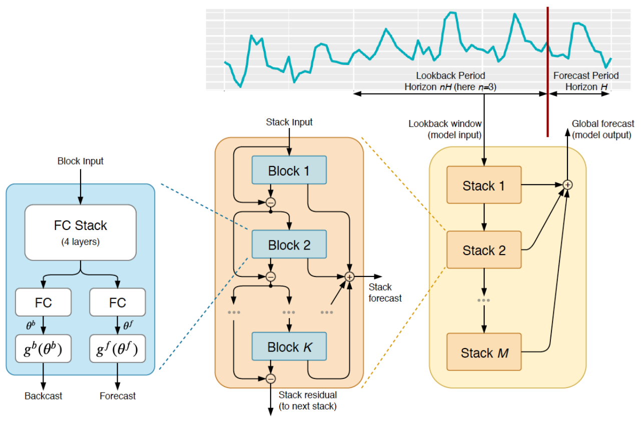


Fig. 6: The architecture of weak learner (N-Beats [8] model) implemented in *EN-Beats* design.

- Assumption1: Data between current time and previous time are correlated and the generated data in near time are the same data structure, i.e., the same dimension and data type.
- Assumption2: The data distribution shows differences as time goes by, i.e., not the same data distribution all the time.
- Assumption 3: Data are generated in a continuous way and the time consumed by model training is longer than the time for data generation, thus the model could be continually trained with new accumulated data.

A. *Framework of EN-Beats*

To learn the behavior of the use of resources that fluctuate continually we propose the *EN-Beats* model with a global strong learner stacked by different weak learners with different weights, as shown in Figure 5. For a basic weak learner, we adopted an N-Beats (Neural basis expansion analysis for interpretable time series forecasting) model is a deep learning model for time series forecasting that was introduced by Oreshkin et al. [8] in 2020. By using ensemble learning, we integrate weak learners into a strong learner, as indicated in Figure 5.

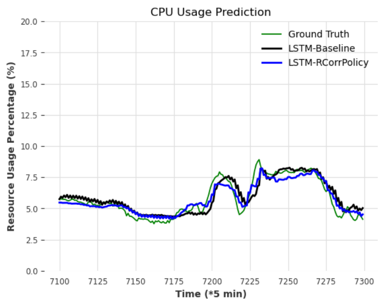
1) Weak learner: The basic building block of *EN-Beats* is the N-Beats-based weak learner (WL). Oreshkin et al. [8] proposed N-Beats model for providing interpretable time-series forecasting. It is based on a stack of fully connected layers, where each layer consists of a set of basis functions that are learned from the data. These basis functions can be thought of as a kind of dictionary of time series patterns, and the model learns how to combine them to make predictions.

To build up the weak learner, N-Beats is chosen as the basic model for the following reasons. Firstly, the N-BEATS model is highly modular and can be easily scaled to handle large and complex time series data and can also be easily adapted to different types of time series data and forecasting tasks. Secondly, it uses a decomposition of the time series into a sequence of basis functions, which makes it highly interpretable and helps to understand the contribution of each component to the forecast and identify trends, seasonal patterns, and other factors that may affect the time series. Thirdly, it has a simple architecture that makes it fast to train compared to other deep learning models. This makes it suitable for applications where speed is important, such as real-time forecasting. Fourthly, it has been shown to achieve state-of-the-art performance on several benchmark time series datasets, outperforming other popular deep learning models like LSTMs and GRUs.

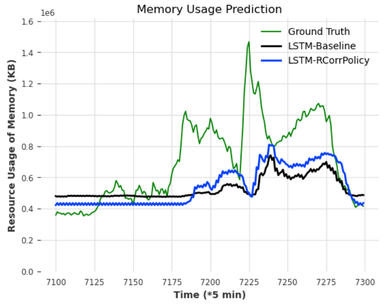
2) Strong learner: Inspired by ensemble learning, we build the *Strong Learner* (SL) from *WL*s with different weights. The aggregation process of *WL*s into SL can be summarized as the following process. Firstly, we partition the original dataset D into m chunks of sub-dataset with equal length, which can be represented by $D = \bigcup_{i=1}^{m} Data_i$. Secondly, we group these sub-dataset into different groups by following $Group_n = \bigcup_{i=1}^{n} Data_i$, in which $1 <= n <= m$. Thirdly, each *WL* will be trained by different groups of data. Fourthly, we train a linear regression model to aggregate the *weak learners* into the *strong learner*. When new data is coming, the newly accumulated data will be added as a new sub-dataset to update the recent data groups and train new weak learners.

VI. Performance Evaluation

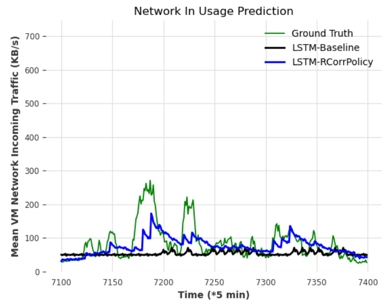
In this section, we describe the performance evaluation for *RCorrPolicy* and *EN-Beats*, including dataset introduction,



(a) LSTM - CPU utilization rate prediction

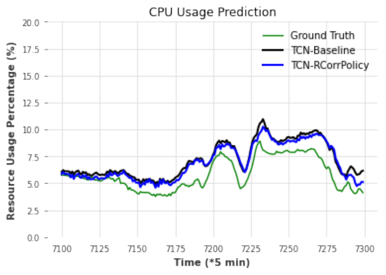


(b) LSTM - Memory usage prediction

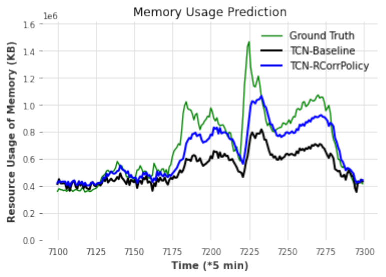


(c) LSTM - Network incoming traffic prediction

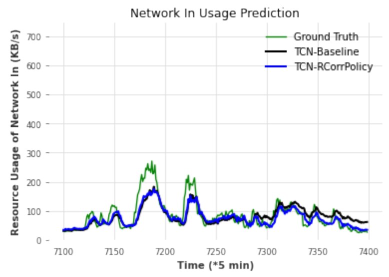
Fig. 7: Resource usage predictions by LSTM baseline model with *RCorrPolicy implemented*



(a) TCN - CPU utilization rate prediction

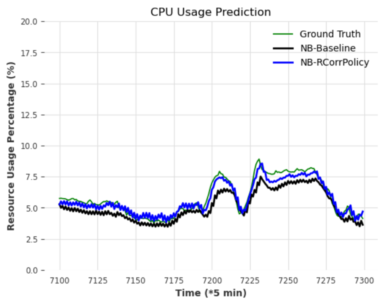


(b) TCN - Memory usage prediction

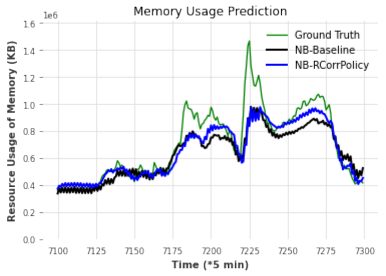


(c) TCN - Network incoming traffic prediction

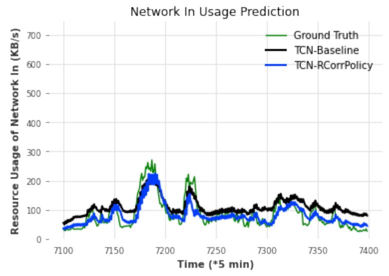
Fig. 8: Resource usage predictions by TCN baseline model with *RCorrPolicy implemented*.



(a) N-BEATS - CPU utilization rate prediction



(b) N-BEATS - Memory usage prediction



(c) N-BEATS - Network incoming traffic prediction

Fig. 9: Resource usage predictions by N-BEATS baseline model with *RCorrPolicy* implemented.

baseline model explanations, experimental setup, ablation experiments for *RCorrPolicy* and evaluations for *EN-Beats* model.

A. Dataset

`Bitbrains` [2] dataset[2] was collected from the cloud of Bitbrains, which is a cloud service provider and specializes in providing financial computation services for enterprise customers, including many major banks, credit card operators, and insurers. In `Bitbrains` fastStorage dataset, it includes workload traces of 1,250 VMs that were used for co-located business software applications within Bitbrains cloud data center. These data are stored as CSV files across 30 days with `5 min` sampling interval. In Table II, the collected resource metrics and the corresponding statistical characteristics are summarized. As observed in Figure 1 and Figure 3, the `Bitbrains` dataset shows the evolving resource usage trends with high dynamics and uncertainties, making the predictions of resource usages difficult.

B. Baseline Models

In the comparison experiments, we evaluated the effectiveness of baseline models with the proposed *RorrPolicy*. There are three models including RNN-based LSTM [9], TCN

[2]The open-source Bitbrains dataset can be downloaded from http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

TABLE II: Statistical characteristics of the GWA-T-12 Bitbrains fastStorage dataset [2] used in our evaluations.

Properties	Mean	Min	Max	SDev
CPU requested [GHz]	8.9	2.4	86	11.1
CPU usage [GHz]	1.4	0.0	64	4.4
Memory requested [GB]	10.7	0.0	511	29.3
Memory usage [GB]	0.6	0.0	384	1.8
Disk, Read throughput [MB/s]	0.3	0.0	1,411	5.2
Disk, Write throughput [MB/s]	0.1	0.00	188	1.1
Network receive [MB/s]	0.1	0.0	859	0.7
Network transmit [MB/s]	0.1	0.0	3,193	1.5
CPU cores	3.3	1	32	4.0

(temporal convolutional neural network) [10] and N-BEATS model [8]. Many researchers have used RNN-based models like LSTM for cloud workload prediction [9] [4], and other deep learning models like TCN [3] and N-BEATS model [8] for tackling new sophisticated scenarios.

1) LSTM model: Long Short-Term Memory (LSTM) is a special form of Recurrent Neural Network (RNN) that could learn long-term dependencies from historical data. The recurring module is achieved through a combination of four layers exchanging information with each other. In a typical LSTM cell, there are three gates (input gate, forgot gate, and output gate) and a timely-changing cell state. The gates and cell state empowers LSTM to selectively learn, unlearn and retain information as time goes by.

2) TCN model: A TCN (Temporal Convolutional Network) model consists of dilated, causal 1D convolutional layers with the same input and output lengths. This model was recently proposed by Lea et al. [10] for modeling sequential data samples with previously ignored convolutional modules. While previous RNN-based models such as LSTM, bi-LSTM, and GRU have demonstrated favorable performance with certain data samples, they often encounter issues with exploding or vanishing gradients, which hampers their overall efficiency. By integrating a convolutional module rather than a recurrent one can improve the overall performance because it allows for parallel computation of outputs. TCN model and its modified version have gained popularity in the recent few years.

3) N-BEATS model: For univariate time series prediction, N-BEATS (Neural Basis Expansion Analysis) for interpretable Time Series" is a type of recently proposed neural network with deep architectures based on backward and forward residual links, with deep stacks of fully-connected layers [8]. The N-BEATS model is capable of making more accurate predictions and interpretable results without sacrificing training time. N-BEATS has attracted increasing attention and is treated as a comparison model by state-of-the-art works [26] [27]. However, the original N-BEATS model is designed for predicting univariate time series, while our scenario is multiple resource usage prediction, thus we made some changes to it to suit the multivariate inputs and outputs.

C. Evaluation Metrics

To evaluate the effectiveness of the proposed methods, we adopted three different performance metrics as follows.

1) MSE (Mean Square Error):

$$MSE(y_i, \hat{y_i}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{4}$$

2) NRMSE (Normalized Root Mean Square Error):

$$NRMSE(y_i, \hat{y_i}) = \frac{1}{y_{max} - y_{min}}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{5}$$

3) R^2 score (Coefficient of determination):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2} \tag{6}$$

In Equations (4), (5), and (6), n is the number of all true values, y_i is the value of truth observations, $\hat{y_i}$ is the prediction value, y_{max} and y_{min} are the maximum and minimum values among all y_i values respectively.

D. Experimental setup

We configured and tuned different parameters for baseline models for evaluating the proposed *RCorrPolicy* policy and *EN-Beats* model. All the algorithms are configured to provide predictions of the target metric two timestamps in advance, which is `10 min` in our scenario because `10 min` prediction in the cloud environment is enough for the system scheduler to make decisions. The threshold in Algorithm 1 for *RCorrPolicy* is configured as `T = 0.7`. In TCN model, some key parameters are set as `input_chunk_length = 8`, `output_chunk_length = 2`, `dropout = 0.08`, `dialation_base = 2`, `kernal_size = 5`, `learning_rate=1e-3`. In LSTM model, the key parameters are set up as `input_chunk_length=8`, `output_chunk_length=2`, `dropout=0.01`, `hidden_dim = 10`, `learning_rate = 1e-3`, `batch_size = 32`. For N-BEATS model, the key parameters are configured as `input_chunk_length = 8`, `output_chunk_length = 2`, `dropout = 0.08`, `learning_rate = 1e-3`, `num_blocks = 4`, `num_stacks = 30`, `expansion_coefficient_dim = 5`, `activation_function = ReLU`. For the EN-beats model, the weak learner's key parameters are the same as those used for N-BEATS.

E. Ablation Experiments for RCorrPolicy

1) RCorrPolicy Effectiveness: We analyzed the correlations among VM metrics and plotted the heatmap, presented in Figure 4. For each resource that we plan to predict, we used the *RCorrPolicy* to select metrics according to our needs and evaluated the effectiveness of the proposed policy. For the comparison experiments, we selected three baseline models, including TCN [10], LSTM [9], N-BEATS [8], which are widely used for time-series prediction problems. For each baseline model, we selected three different metrics (i.e., CPU utilization rate, Memory utilization, and Network incoming traffic) in experiments and the quantified experiment results are summarized in Table III.

TABLE III: The evaluation of **RCorrPolicy** with different models for predicting different resource metric types with evaluation metrics including MSE, NRMSE, and R^2.

COMPARISON		Evaluation Metrics					
		Without RCorrPolicy			With RCorrPolicy		
Baseline models	Resource metrics	MSE	NRMSE	R^2	MSE	NRMSE	R^2
LSTM [9]	CPU util. rate	0.44	4.8%	0.79	0.24	3.5%	0.89
	Memory usage	$83.19*10^9$	20%	-0.23	$67.69*10^9$	18.49%	-0.01
	Network_in	3669.55	8.2%	-0.38	1781.82	5.7%	0.33
TCN [10]	CPU util. rate	2.2	10%	-0.04	1.68	9.4%	0.20
	Memory usage	$55.74*10^9$	16%	0.17	$17.04*10^9$	9.2%	0.74
	Network_in	1163.26	4.6%	0.56	915.43	4.1%	0.65
N-BEATS [8]	CPU util. rate	0.68	5.9%	0.68	0.15	2.8%	0.93
	Memory usage	$17.41*10^9$	9.4%	0.74	$15.48*10^9$	8.6%	0.78
	Network_in	1548.45	5.3%	0.42	885.75	4.1%	0.67

2) Discussions of RCorrPolicy evaluation: In Figure 7, Figure 8, and Figure 9, we present the ablation experiments by implementing three models for predicting CPU utilization rate, Memory usage, and Network incoming traffic resource metrics. It can be observed that models implemented with *RCorrPolicy* outperform the corresponding models that do not include *RCorrPolicy*. From the experimental results in Figure 8b, the results demonstrate a notable reduction in `NRMSE` from 16% to 9.2%, representing a substantial improvement of 44.7%. In addition, from the experimental results shown in Table III, it can be observed that TCN and N-BEATS models show better overall performance in predicting different resource metrics. Furthermore, when comparing TCN and N-BEATS models, according to their designed model structures [10] [8], TCN demonstrates suitability for long-term time series sequences, while the N-BEATS model excels at capturing both short-term and long-term time series patterns, making it particularly valuable in dynamic and uncertain cloud environments.

F. Evaluation of EN-Beats model

We evaluated the proposed *EN-Beats* model for resource metric prediction alongside baseline models. Figure 10, Figure 11, and Figure 12 demonstrate the superior performance of the proposed *EN-Beats* in comparison to the baseline models.

1) Discussion of experimental results: To evaluate the effectiveness of the proposed *EN-Beats*, we conducted extensive experiments comparing it with existing models in terms of multiple resource predictions for resource usages from `Bitbrains` [2]. As depicted in Figure 10, Figure 11, and Figure 12, *EN-Beats* demonstrates strong performance across various resource metric predictions, exhibiting lower *MSE* (lower values indicate better performance), lower *NRMSE* (lower values indicate better performance), and higher R^2 score (higher values indicate better performance). Therefore, *EN-Beats* outperforms the existing models.

For the prediction of CPU utilization rate, Figure 10 illustrates that our proposed *EN-Beats* model closely follows the trend of the ground truth, outperforming other models. In terms of *NRMSE*, the LSTM model, TCN, N-BEATS, and EN-Beats achieved respective experimental results of 12%, 6.5%, 5.8%, and 4.4%. Thus, the proposed *EN-Beats* exhibits lower prediction errors for the CPU utilization rate. Additionally, our method attains the highest R^2 score, indicating superior generalization ability compared to the other methods under consideration.

TABLE IV: The evaluation of **EN-Beats** model and baseline models for predicting CPU utilization.

CPU util. rate (%)	MSE	NRMSE	R^2
LSTM [9]	2.92	12.4%	-0.56
TCN [10]	0.80	6.51%	0.62
N-BEATS [8]	0.64	5.84%	0.66
EN-Beats(This work)	0.36	4.4%	0.83

In terms of resource prediction for Memory usage, Figure 11 and Table V demonstrate that *EN-Beats* effectively captures the changing patterns of the ground truth. Quantitatively, the NRMSE values for LSTM, TCN, N-BEATS, and EN-Beats are 13.97%, 11.82%, 8.22%, and 5.93%, respectively. These results indicate that our proposed method exhibits the lowest error rate and highest accuracy. Furthermore, our method achieves a higher R^2 score (indicating better performance) compared to other methods, further highlighting the superiority of our trained model.

For predicting `network` incoming traffic, Figure 12 demonstrates the close estimation achieved by our proposed *EN-Beats* model compared to the ground truth. Regarding the NRMSE values of different models, Table VI presents the normalized errors for LSTM, TCN, N-BEATS, and EN-Beats as 6.37%, 6.78%, 4.50%, and 6.44%, respectively. Although the NRMSE of our proposed model is slightly higher than that of other models, it is important to note that our model excels in another crucial evaluation metric, the R^2 score, indicating its ability to effectively fit the data.

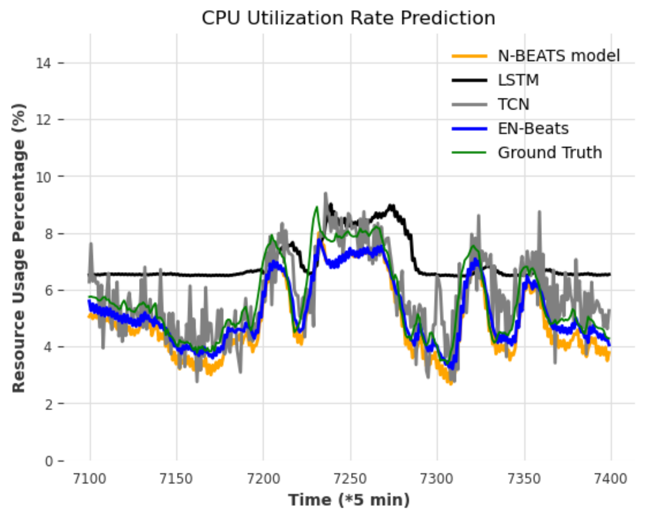


Fig. 10: A comparison of CPU utilization rate predictions.

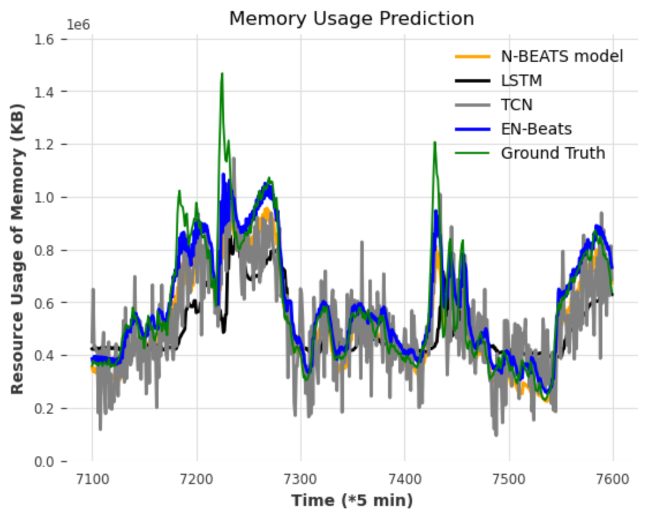


Fig. 11: A comparison of memory usage predictions.

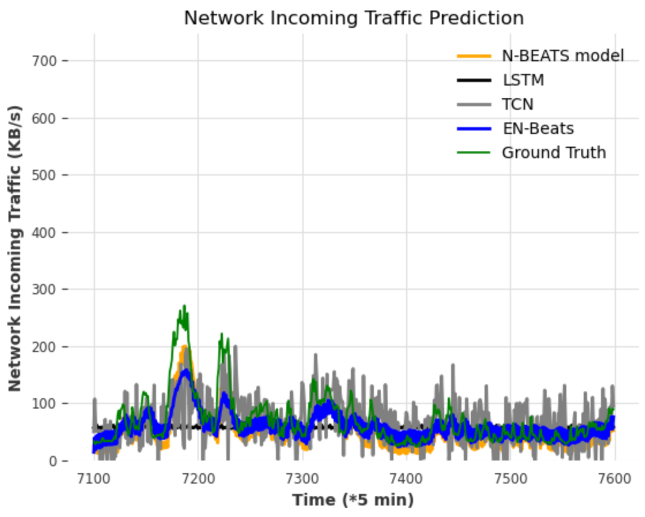


Fig. 12: A comparison of network incoming traffic predictions

TABLE V: The evaluation of **EN-Beats** model and baseline models for predicting memory utilization.

Memory usage (KB)	MSE	NRMSE	R^2
LSTM [9]	$38.56*10^9$	13.97%	0.31
TCN [10]	$27.65*10^9$	11.82%	0.51
N-BEATS [8]	$13.37*10^9$	8.22%	0.76
EN-Beats(This work)	$6.96*10^9$	5.93%	0.88

TABLE VI: The evaluation of **EN-Beats** model and baseline models for predicting network incoming traffic.

Network_in traffic (KB/s)	MSE	NRMSE	R^2
LSTM [9]	$21.79*10^2$	6.37%	-0.08
TCN [10]	$24.74*10^2$	6.78%	-0.22
N-BEATS [8]	$10.85*10^2$	4.50%	0.46
EN-Beats(This work)	$22.02*10^2$	6.40%	0.54

VII. Conclusions and Future Work

In this paper, we introduce two novel contributions: an efficient metric selection policy called *RCorrPolicy* and an ensemble learning-based model called *EN-Beats* for multiple resource predictions. Our performance evaluation demonstrates that the proposed *RCorrPolicy* approach enhances prediction accuracy for three key metrics (CPU utilization rate, memory usage, and network incoming traffic) across different algorithms (LSTM, TCN, N-BEATS) through effective feature engineering. Furthermore, our proposed *EN-Beats* model showcases the ability to predict computing resource usage with 10-minute advance notice. In terms of R^2 score, a metric indicating the goodness-of-fit between the data and trained models, our experiments yield promising results, with 0.83%, 0.88%, and 0.54% for CPU utilization rate, memory usage, and network incoming traffic metrics, respectively. Overall, our proposed methods outperform existing approaches across various evaluation metrics when predicting multiple resource metrics.

Moving forward, our future work will focus on automatic determination of the threshold parameter T for *RCorrPolicy*, as well as integration of the proposed multiple resource prediction methods into Virtual Machine (VM) auto-scaling approaches. By achieving more accurate predictions of future resource usage, the scheduler can dynamically adjust VM provisioning, scaling up when resources are over-utilized and scaling down during periods of low utilization. This enables cloud providers to optimize their systems while delivering consistent Quality of Service. Additionally, we plan to evaluate our approach using more workload datasets from large-scale clouds such as Alibaba and Google Cloud.

Acknowledgments

This work is supported by Melbourne Research Scholarship and China Scholarship Council. We thank Leila Fayez Ismail for comments on improving the paper.

References

[1] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become: The next phase of cloud computing," *Commun. ACM*, vol. 64, p. 76–84, apr 2021.

[2] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 465–474, 2015.

[3] W. Chen, C. Lu, K. Ye, Y. Wang, and C.-Z. Xu, "Rptcn: Resource prediction for high-dynamic workloads in clouds based on deep learning," in *Proceedings of the 2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 59–69, 2021.

[4] M. Xu, C. Song, H. Wu, S. S. Gill, K. Ye, and C. Xu, "Esdnn: Deep neural network based multivariate workload prediction in cloud computing environments," *ACM Trans. Internet Technol.*, vol. 22, aug 2022.

[5] F. Farahnakian, P. Liljeberg, and J. Plosila, "Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers," in *Proceedings of the 2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 357–364, 2013.

[6] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *Proceedings of the 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, 2020.

[7] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[8] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *Proceedings of the International Conference on Learning Representations*, 2020.

[9] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018. The 6th International Conference on Smart Computing and Communications.

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

[11] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and arima model in cloud data center," in *Proceedings of the 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6, 2018.

[12] Y. Wang, C. Wang, C. Shi, and B. Xiao, "Short-term cloud coverage prediction using the arima time series model," *REMOTE SENSING LETTERS*, vol. 9, no. 3, pp. 274–283, 2018.

[13] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, pp. 500–507, 2011.

[14] J. Bi, H. Yuan, and M. Zhou, "Temporal prediction of multiapplication consolidated workloads in distributed clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1763–1773, 2019.

[15] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos, "Adjusted fair scheduling and non-linear workload prediction for qos guarantees in grid computing," *Computer Communications*, vol. 30, no. 3, pp. 499–515, 2007. Special Issue: Emerging Middleware for Next Generation Networks.

[16] G. K. Shyam and S. S. Manvi, "Virtual resource prediction in cloud environment: A bayesian approach," *Journal of Network and Computer Applications*, vol. 65, pp. 144–154, 2016.

[17] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for vm workload prediction in the cloud," in *Proceedings of the 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 319–324, 2016.

[18] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, (New York, NY, USA), p. 153–167, Association for Computing Machinery, 2017.

[19] C. Nguyen, C. Klein, and E. Elmroth, "Multivariate lstm-based location-aware workload prediction for edge data centers," in *Proceedings of the 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 341–350, 2019.

[20] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, 2021.

[21] M. E. Karim, M. M. S. Maswood, S. Das, and A. G. Alharbi, "Bhyprec: A novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine," *IEEE Access*, vol. 9, pp. 131476–131495, 2021.

[22] D. Janardhanan and E. Barrett, "Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models," in *Proceedings of the 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 55–60, 2017.

[23] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, pp. 191–201, 2015.

[24] A. A. Bankole and S. A. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 156–161, 2013.

[25] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A quantum approach towards the adaptive prediction of cloud workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 12, pp. 2893–2905, 2021.

[26] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, 2021.

[27] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "Meta-learning framework with applications to zero-shot time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9242–9250, 2021.