Maiyza *et al. Journal of Cloud Computing* (2023) 12:97
https://doi.org/10.1186/s13677-023-00473-z

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

VTGAN: hybrid generative adversarial networks for cloud workload prediction

Aya I. Maiyza[1,2*], Noha O. Korany[1], Karim Banawan[1], Hanan A. Hassan[2] and Walaa M. Sheta[2]

Abstract

Efficient resource management approaches have become a fundamental challenge for distributed systems, especially dynamic environment systems such as cloud computing data centers. These approaches aim at load-balancing or minimizing power consumption. Due to the highly dynamic nature of cloud workloads, traditional time series and machine learning models fail to achieve accurate predictions. In this paper, we propose novel hybrid VTGAN models. Our proposed models not only aim at predicting future workloads but also predicting the workload trend (i.e., the upward or downward direction of the workload). Trend classification could be less complex during the decision-making process in resource management approaches. Also, we study the effect of changing the sliding window size and the number of prediction steps. In addition, we investigate the impact of enhancing the features used for training using the technical indicators, Fourier transforms, and wavelet transforms. We validate our models using a real cloud workload dataset. Our results show that VTGAN models outperform traditional deep learning and hybrid models, such as LSTM/GRU and CNN-LSTM/GRU, concerning cloud workload prediction and trend classification. Our proposed model records an upward prediction accuracy ranging from 95.4% to 96.6%.

Keywords Cloud computing, Workload prediction, GAN, LSTM, GRU, Convolution neural network, sliding windows, Multi-step-ahead-prediction

Introduction

Recently, there has been a pronounced tendency towards using individual virtual servers in large-scale cloud data centers with thousands of high-performance servers. For instance, cloud services provide elastic computing advantages to end users based on virtualization technology at a low-cost [16, 68]. Virtual machine (VM) facilities allow cloud end users to scale up/down or relinquish their resource demands (e.g., CPUs/GPUs, memory, storage, $\cdots$, etc.) and pay accordingly. Such frequent variations in the dynamic environment lead to a tradeoff between the service provider's profit and the end user's quality of service (QoS). More specifically, the underutilized server causes resource and power consumption wastage. On the other hand, the overutilized server causes performance degradation. Consequently, service providers need efficient techniques for optimal resource management [33, 68]. Managing and improving the provided services in such distributed systems cause several challenges. One major challenge is observing and monitoring these distributed systems for accurate resource allocation decisions [58]. In particular, observability has become a critical prerequisite to guarantee stable services for end-user applications and maximize the profit for the service provider.

In general, there are two approaches for resource allocation: reactive and proactive [77]. The *reactive* approach offloads the required resources from overutilized servers to underutilized servers. The offloading decisions, in this case, rely on the *current* end-user utilization. Nevertheless, this causes unnecessary migration because of the

*Correspondence:
Aya I. Maiyza
amaiyza@srtacity.sci.eg
[1] Department of Electrical Engineering, Faculty of Engineering, Alexandria University, Alexandria, Egypt
[2] Informatics Research Institute, City of Scientific Research and Technological Applications (SRTA-City), Alexandria, Egypt

sharp workload peaks. Hence, researchers exert continuous effort to improve the accuracy of *proactive* resource allocation techniques, where deciding VM migration depends on *future* workloads [71]. Most researchers focus on predicting CPU utilization for the servers [24, 54], or individual VMs [55]. The motivation for focusing on CPU utilization stems from the fact that the CPU of a server incurs the most power consumption, and the relationship between energy consumption and CPU utilization is linear [15].

Focusing on the proactive resource allocation approaches, we need an accurate forecasting technique. To that end, classical time-series techniques aim to model short-term forecasts. As the CPU utilization data is considered time series data, The ARIMA models have been widely used for CPU utilization time series forecasting [57]. For example, researchers have used ARIMA models as a baseline to compare more sophisticated techniques [41]. The main drawback of the time series forecasting model is that it merely captures linear relationships. In addition, TS models require the input data to be stationary (whether in its raw form or as differenced data). Unfortunately, authors in [55] performed the popular Kwiatkowski-Phillips-Schmidt-Shin (KPSS) stationarity test for each VM [40]. They concluded that almost 70% of tested PlanetLab VMs [60] are *not stationary*. Consequently, classical TS models cannot accurately predict its future CPU utilization. As a result, they used machine learning (ML) models to predict the CPU utilization using lagged values of each time series as inputs to the model. Hence, in recent years many machine learning models, such as artificial neural networks (ANNs) [55, 66, 67], and support vector machine (SVM) [6, 37, 55], have been proposed for modeling CPU utilization.

Deep learning (DL) methods have stirred remarkable attention during the artificial intelligence revolution in recent years. Deep-learning-based prediction models outperform traditional machine learning models in several applications, especially cloud workloads prediction [48]. Thus, the accuracy of CPU utilization prediction could increase using a recurrent neural network (RNN), which maps target vectors from the history of the previous inputs. Nevertheless, RNN suffers from the gradient vanishing problem with long sequences [57]. The long short-term memory (LSTM), which Hochreiter and Schmidhuber [35] proposed, is an effective solution to overcome the gradient vanishing problem. LSTM achieves a considerable improvement in capturing long-term temporal dependencies. Thus, LSTM can accurately predict high fluctuated time-series data [59, 76]. Recently, the generative adversarial network (GAN), proposed by Goodfellow [30], achieves remarkable improvements in different research areas. In particular, GANs are used for the prediction of highly volatile cloud traces as in [85]. This motivates our interest in investigating the performance of GANs for workload prediction. GANs employ two deep learning networks, namely, the generator and the discriminator. The generator generates artificial data samples that mimic the actual distribution of the actual data distribution. The discriminator, however, tries to differentiate between the actual data samples and the artificially generated samples by the generator. By providing a feedback signal from the discriminator to the generator, the generator enhances its data generation model.

Moreover, many research works concerning forecasting investigated the problem of selecting technical indicators (TIs) as input of machine learning/deep learning models for extracting more features [74]. Many efforts study the determination of the optimal combinations of TIs or their parameters.

The main challenge in cloud prediction is the need for an effective nonlinear model that tracks the cloud workload [45, 79]. Furthermore, the workload value frequently suffers from excessive changes [62]. This motivates our interest in recasting the over-utilized server detection problem into a workload *trend* prediction rather than the value. In other words, the system will migrate VMs from over-utilized servers if the future workload trend is "up" only. We inspire this idea from stock price prediction, where researchers in this area demonstrated that trend prediction as a classification problem can improve prediction accuracy using machine learning and deep learning models [23, 70].

Therefore, the principal contribution of this paper is proposing a novel nonlinear prediction model, named value trend generative adversarial network (VTGAN), to deal with the high-frequency and volatility of cloud workload. Additionally, this paper presents a novel classification approach to predict the trend of workload data. In our proposed VTGAN prediction model, we used a GAN in which the long short-term memory (LSTM) or the gated recurrent units (GRU) model is a generator, and the convolution neural network (CNN) model is a discriminator. The proposed system presents subsequent research contributions:

- We use GAN models for building predicting cloud workloads models. Moreover, GANs were not applied before in cloud data centers, whether a simulation or real environment, making our model one of the pioneers in cloud workload prediction.
- In addition, we compared the results of the proposed models with state-of-the-art time series, ML, and DL models, such as ARIMA, SVR, LSTM, and GRU.

- We propose a classification approach to predict the trend instead of the value of the cloud workload.
- We study the effect of using common technical indicators.
- We also study and test the window input size and multi-step prediction using our model.

The structure of this paper is as follows: Section "Related work" presents the related work. Section "Proposed architecture" introduces the mathematical model. Section "Experimental configuration and evaluation methodology" shows the experimental set-up and the methodology of the evaluation conducted in this work. Section "Results and discussions" analyzes the performance results. Section "Conclusions and future works" summarizes our concluding remarks.

Related work

During the last decade, machine learning and deep learning approaches have revolutionized the scientific and industrial communities. In the sequel, we focus on enumerating research works concerning the time-series prediction area. Figure 1 illustrates a taxonomy of time-series prediction models. Classically, most works deal with workload forecasting as a value prediction problem (a.k.a. regression). We classify the regression models into four main categories: (i) Traditional time series models, (ii) Machine Learning models, (iii) Deep learning models, and (iv) Hybrid Techniques. Nevertheless, in this work, we will introduce a trend prediction approach (a.k.a. classification), where we focus on predicting the sign of workload change.

Traditional time series approaches

As cloud workload data is naturally temporal, researchers used different time-series forecasting models for predicting workload traces. Autoregressive moving average (ARMA), as a traditional time-series forecasting model, is used in [17] to predict cloud workload for resource allocation. Authors reported that this approach is unsuitable for most cloud workload traces, particularly for highly-volatile workloads. Also, Vazquez et al. [81] applied several time-series prediction models, such as AR, MA, simple exponential smoothing (SES), double exponential smoothing (DES), error trend seasonal exponential smoothing (ETS), and ARIMA, to forecast cloud workloads. They evaluated the forecasting accuracy for each model for two real cloud workloads, namely, Google cluster data and Intel Netbatch logs. The authors conclude that no model is consistently superior to the others for all datasets.

Vashistha and Verma [80] presented a cloud workload prediction survey based on time series models, where some researchers applied AR [37–39, 46], MA [37, 38, 81], and ARIMA [7, 17, 18, 28, 38, 46, 81]. In addition, other researchers proposed extended versions of the ARIMA model for workload prediction, such as autoregressive moving average with exogenous inputs (ARMAX) [88], cumulative moving average (CMA), weighted moving average (WMA) [29], difference model (DM), and median model (MM) [38].

Although such traditional time-series approaches were ubiquitous in the last decade, these models are not appropriate for long-term time-series data [47]. Moreover, these models assume that the input data is stationary, which is not a valid assumption for most cloud workload traces [55]. Therefore, the ML approaches seem like a natural solution for traditional time-series problems and a step toward more accurate cloud workload prediction results.

Machine learning approaches

ML models have been widely used as an alternative solution for traditional time-series forecasting. Thus, researchers proposed several ML prediction models for cloud applications. Farahnakian et al. [25] proposed a linear regression (LR) algorithm to predict the CPU utilization of the servers in the context of proactive overload detection servers. In follow-up work, they used a *K*-nearest neighbor (KNN) regression model instead of the linear regression model. They demonstrated that this approach is superior in terms of energy consumption and system performance [26].

Patel et al. [63] proposed the support vector regression (SVR) and ARIMA models to predict VM memory during the live migration to calculate the migration time. The SVR model has less capability to improve prediction accuracy because it consists of a single hidden layer. Cortez et al. [21] used gradient boosting tree and random forest models to predict the resource management of a VM allocated in the Azure cloud platform. They used the dynamically linked library (DLL) to collect the result after each estimation process. Then, it decided whether the prediction process was trusted using the DLL score.

Nguyen et al. [34] used a multiple linear regression (MLR) method to predict overutilized and underutilized servers. They integrated their prediction technique with traditional consolidation frameworks to reduce energy consumption.

Moghaddam et al. [55] proposed different ML algorithms for overload detection in the VM consolidation framework. They developed several ML prediction

algorithms for individual VMs to predict the most suitable time for migration from overutilized servers. They implemented their approach using PlanetLab traces based on the CloudSim simulation tool [60]. Their framework was compared to LR-MMT-PBFD as a baseline in most publications. Nevertheless, they did not measure the prediction accuracy of the proposed ML models and implemented them directly on the VM consolidation framework. Thus, in this paper, we evaluate the accuracy of our approaches before integrating them with the whole system in future work.

Regardless of the reasonably fast prediction ability for cloud workloads, ML approaches do not achieve high prediction accuracy with high dispersal because of the non-linearity and complexity of cloud workloads. Hence, the third direction was deep learning (DL) approaches to achieve high prediction accuracy.

Deep learning approaches

Due to the recent success of DL in various applications, several works employed DL approaches for time-series analysis and prediction [27]. Specifically, the recurrent neural network (RNN) has outstanding sequential processing capabilities. Therefore, authors in [24, 36, 87] proposed an RNN-based model to predict the future workloads in cloud data centers. However, previous research showed that traditional RNNs struggle to capture long-term dependencies due to the vanishing gradient problem [14, 82]. To solve this issue, LSTM [31] and GRU [20] were developed for better dealing with long-term dependencies [19, 42]. Consequently, Song et al. [76] used the LSTM network for workload prediction to improve their previous RNN-based work [84]. GRU is much less computationally intensive than LSTM due to its ability to converge with fewer parameters [20]. Nevertheless, there is little research work based on GRU networks [19, 32] for workload prediction in the cloud environment.

Focusing on convolutional neural networks (CNNs), Mozo et al. [56] used CNN to predict short-term network traffic in data centers. [56] is considered the only work using a pure CNN approach for prediction in the cloud environment because CNN is also unsuitable for long-term dependencies. That is because CNN models fundamentally focus on extracting features and interdependencies from the input sequence and do not use any historical data during the learning process [69].

The nature of cloud workloads is always dynamic and complex. Thus, all previous approaches did not achieve acceptable prediction accuracy due to the long-term dependencies, complexity, and non-linearity of cloud workload traces. As a result, the authors recently tuned the research direction to hybrid approaches rather than single models.

Hybrid approaches

Finally, the hybrid approaches are an amalgamation of various time-series algorithms aiming at forecasting complex time series traces [85]. Liu et al. [52] proposed a hybrid prediction model that combines ARIMA with LSTM models. Their results illustrated that their model improved the prediction accuracy by 6% and 66% compared to the pure LSTM and pure ARIMA models, respectively. Also, Shuvo et al. [73] proposed a hybrid prediction model, namely LSRU, that combined the GRU with the LSTM model. They show that LSRU achieves better accuracy than the pure LSTM or GRU model. Bi et al. [13] proposed a hybrid prediction model integrating bi-directional and grid-long short-term memory networks (BG-LSTM) for high accuracy.

The combination of ConvNets and LSTM is one of the popular hybrid schemes for time series prediction purposes [85]. Regarding cloud environments, Ouhame et al. [59] proposed a hybrid prediction model that combines CNN model with the LSTM model. This combination helps to extract complex features of the VM usage components. This is in addition to modeling temporal information of irregular trends, which may arise in the time series. Their results illustrated that this hybrid model is more accurate than VAR-MLP, VAR-GRU, and ARIMA-LSTM hybrid models.

Recently, the GAN invention revolutionized DL. It achieves remarkable improvement in several fields, such as computer vision and audio. Goodfellow et al. developed GANs in 2014 [30]. Until now, few works considered GAN for time-series cloud workload prediction purposes. The first approach for cloud workload prediction value, E2LG, was proposed by Yazdanian and Sharifan [85]. They combined LSTM networks as a generator and CNNs as a discriminator. This hybrid model can effectively capture the long-term nonlinear dependencies of time series and is suitable for the high-frequency data type. E2LG improved prediction accuracy significantly in the cloud environment. Also, Lin et al. [51] proposed a GAN-based method for realistic cloud workload generation to capture the data distribution and generate high-quality workloads. Generated workloads are useful to mimic real data. In addition, their model can easily generate specific kinds of workloads according to the input. But, their model aimed to generate synthetic data that have a similar distribution to the real data. Unlike our approach, We aim to predict the near future utilization by considering the near historical data to deal with the unexpected change instantaneously.

Table 1 summarizes publications on previous cloud workload prediction approaches. These publications are classified according to their learning category, method, dataset, and weakness.

In this paper, we use a modified version of GAN to predict the trend rather than the value. Therefore, the decision of resource allocation will be based on the trend. This approach is a pioneer in cloud workload prediction. Also, we study the effect of using technical indicators (TIs), Fourier, and wavelet transforms in the performance of our regression and classification models.

Proposed architecture

We propose a modified version of GAN to predict future workload values. The proposed model is a step towards a proactive overload detection technique in the resource management framework for cloud data centers. This technique prevents unnecessary migrations by making migration decisions from the over-utilized server based on the predicted CPU utilization value. In addition, we present an alternative solution to make the migration decision based on the future trend of the cloud workload. For this trend prediction, we cast the prediction problem as trend classification (in contrast to the regression problem corresponding to the workload value prediction).

In our suggested workload prediction system, we use a GAN network. In our proposed GAN architecture, the GRU or LSTM model represents a generator, which learns to generate workload values that are consistent with the statistical distribution of the actual workload. In addition, our GAN model includes a 1D-CNN model as a discriminator, which learns to differentiate between actual and artificially generated workloads. Upon interaction between the generator and discriminator, the predicted workload accuracy enhances. The LSTM and GRU are suitable for predicting time series data. To further enhance the prediction accuracy in multi-step-ahead prediction, our proposed system uses technical indicators (TIs) as feature extraction mechanisms. Moreover, we apply and test Fourier and wavelet transform functions as additional TIs that remove redundant data.

Data preprocessing

To improve the predictive performance of our model, we pre-process the data to highlight oscillations and trends in the workload trace. To that end, we study the use of seven technical indicators (TIs) as additional features. We note that the works [9] and [22] used a subset of these TIs. we extend some of the TIs in [43] to include short-term and long-term moving averages (MAs). These MAs smooth

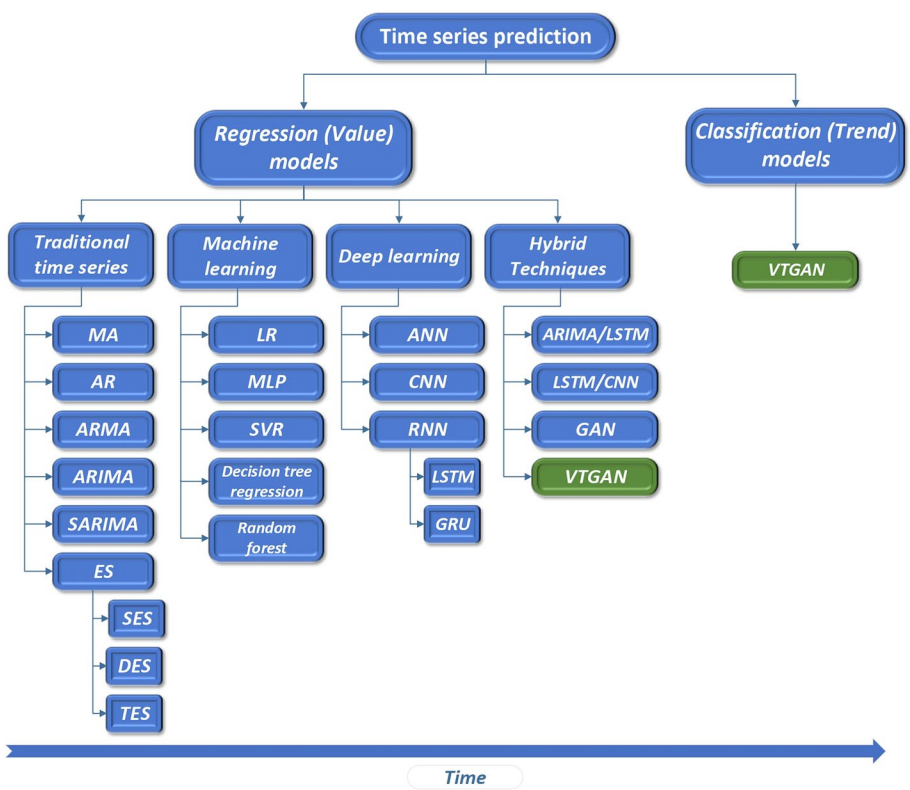


Fig. 1 Taxonomy of cloud workload prediction models

Table 1 Comparison of cloud workload prediction models

	Authors	Method	Dataset	Weakness
Time-series	Calheiros et al. [17]	- ARMA	- Wikimedia Foundation real traces [5]	- Time-series models are not suitable for high volatile workloads, and there is no superior model for all tested datasets.
	Vazquez et al. [81]	AR, MA, SES, DES, ETS, and ARIMA	- Google [3] - Intel Netbatch logs	- These models could not fit with long-term time-series data.
	Kim et al. [46]	AR, ARMA, ARIMA, EMA, DES, WMA, and Gaussian-DES	- Synthetic workloads: Growing & On/Off & Bursty & Random	
	Hu et al. [38]	MA, AR, ARIMA, DM, and MM	- 30 min. from esc.tl.small instance	
	Fu and Zhou [28]	- ARIMA	- PlanetLab [4] - Google	
	Aldossary et al. [7]	- ARIMA	- Collected from OpenNebula testbed	
	Gai et al. [29]	WMA, CMA, MA	-	
	Zhu and Agrawal [88]	- ARMAX	-	
Machine learning	Farahnakian et al. [25]	- LR	- Random workload - PlanetLab	- ML models did not achieve high prediction accuracy with high dispersal data.
	Farahnakian et al. [26]	- KNN		- These models could not fit with non-linear and complex data as cloud workloads.
	Patel et al. [63]	- SVR	- Idle workload - Web workload - Stress workload	
	Cortez et al. [21]	- Gradient boosting tree - Random Forest	- Azure workload	
	Nguyen et al. [34]	- MLR	- Google - PlanetLab	
	Moghaddam et al. [55]	LR, MLP, SVR, AdaBoost, Random Forest, Gradient Boosting, Decision Tree	- PlanetLab	
Deep learning	Zhang et al. [87]	- RNN	- Google	- DL models did not achieve acceptable prediction accuracy due to very long-term dependencies, complex, and non-linearity of cloud data.
	Duggan et al. [24]	- RNN	- PlanetLab	
	Huang et al. [36]	- RNN-LSTM	- Real requests data	
	Yang et al. [84]	- Echo state network (ESN)	- Google	
	Song et al. [76]	- LSTM	- Google	
	Chen et al. [19]	- Auto-Encoder GRU	- Google - Alibaba traces [1]	
	Peng et al. [64]	- GRU based encoder-decoder network	- Google - Dinda [2]	
	Zhu et al. [89]	- Attention-based LSTM	- Alibaba traces - Dinda	
	Mozo et al. [56]	- CNN	- ONTS dataset	
Hybrid	Liu et al. [52]	- ARIMA-LSTM	- Google	- Although its accuracy with non-linearity and very long-term dependencies, it is more complex.
	Shuvo et al. [73]	- LSTM-GRU (LSRU)	- Bitbrains [10]	
	Bi et al. [13]	- BG-LSTM	- Google	
	Ouhame et al. [59]	- CNN-LSTM	- Bitbrains	
	Yazdanian and Sharifan [85]	- GAN (LSTM-CNN)	- Calgary - NASA - Saskatchewan	
	BHyPreC [44]	- Bi-LSTM	- Bitbrains	
	VTGAN	- GAN (Bi-GRU-CNN) - GAN (Bi-LSTM-CNN)	- PlanetLab	

the workload trace, discard short-term fluctuations, and highlight overall trends and/or cycles of the workload time series. In the sequel, we enumerate the full list of our proposed TIs:

- **Moving averages (MAs):** MAs often capture trends by smoothing a CPU utilization series using a lag factor of order n. The long MAs indicators illustrate changes in CPU utilization that are less sensitive to recent utilization movements than the short MAs. This is due to the fact that the longer the MA is, the smoother and less accurate the output is. We calculate MA by Eq. (1), where p_t is the CPU utilization value at time t.

$$MA(p_t, n) = \frac{p_t + p_{t-1} + \cdots + p_{t-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i} \quad (1)$$

- **Exponential Moving Average (EMA):** EMA is a particular moving average indicator, which exponentially averages historic CPU utilization. Unlike simple MAs, EMA can place more weight on recent CPU utilization. More specifically, the influence of previous CPU utilization samples decreases exponentially fast in the EMA indicator. Hence, it reflects directly on the immediate trend [22]. We calculate EMA according to (2),

$$EMA(p_t, s) = \frac{p_t + \alpha p_{t-1} + \cdots + \alpha^t p_0}{1 + \alpha + \cdots + \alpha^t} \quad (2)$$

where s is a tuning parameter to control the importance of the recent past, and α is a weighting term ($\alpha = \frac{s-1}{s+1}$).

- **Moving Average Convergence Divergence (MACD):** It gives insight into workload convergence, divergence, and crossover [22]. It reflects the difference between a short-term (fast) EMA and a long-term (slow) EMA, capturing the second derivative of a CPU utilization series. We calculate MACD according to (3),

$$MACD(p_t, s_1, s_2) = EMA(p_t, s_1) - EMA(p_t, s_2), \quad s_2 > s_1 \quad (3)$$

- **Moving Standard Deviation (MSD):** MSD measures the nth time slot volatility (i.e., the rate of change) of CPU utilization. It is considered helpful in predicting the magnitude of future CPU utilization changes. This indicator expects low-volatility periods followed by high-volatility periods. We calculate MSD according to (4),

$$MSD(p_t, n) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(p_{t-i} - MA(p_t, n))^2} \quad (4)$$

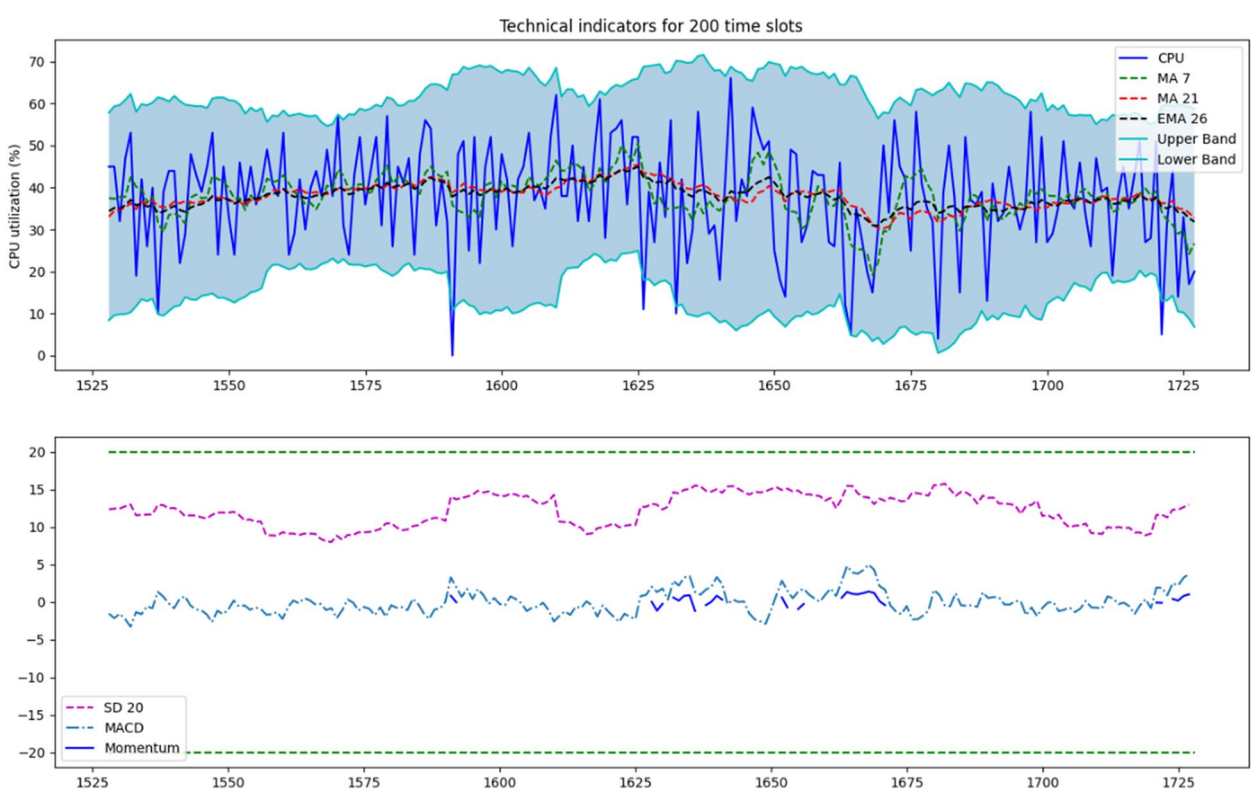


Fig. 2 Selected technical indicators after applied to the used dataset. (200-time slots)

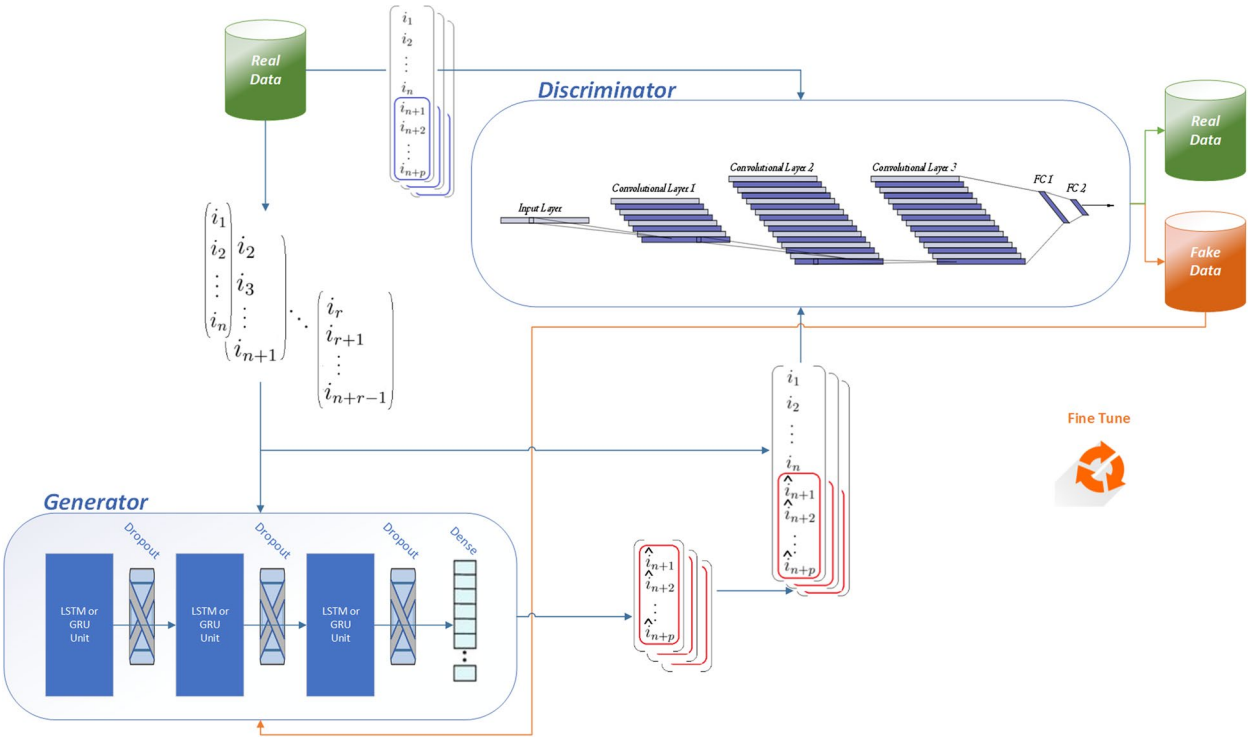


Fig. 3 The proposed VTGAN architecture

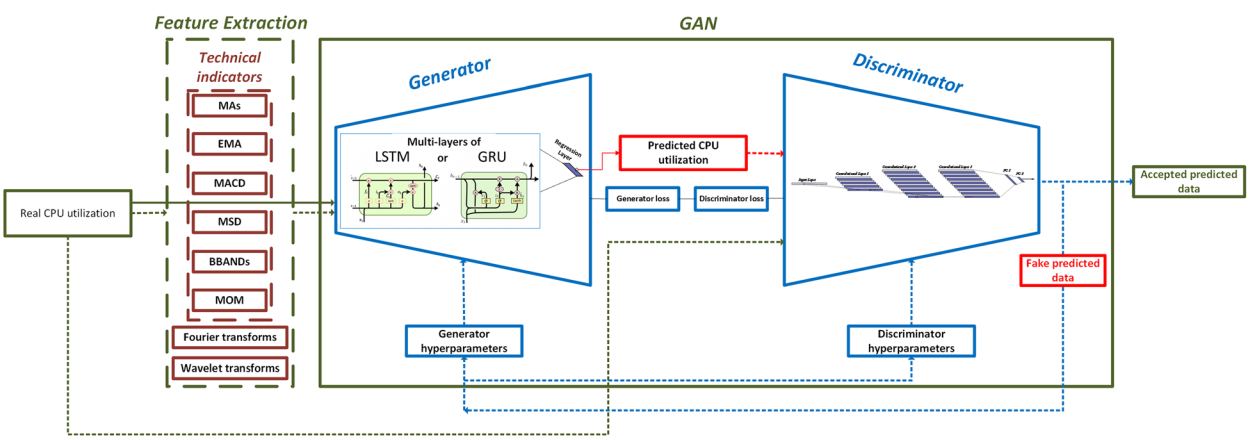


Fig. 4 The proposed VTGAN model

- **Bollinger Bands (BBANDs):** Bollinger Bands are indicators that are plotted at standard deviation levels above, and below a simple moving average. BBANDs consist of the upper band ($BBAND^{+}$) and the lower band ($BBAND^{-}$) [22]. Bollinger Bands are useful indicators to compare volatility against relative CPU utilization levels, over a period of time. We calculate $BBAND^{+}$ and $BBAND^{-}$ by Eqs. (5) and (6).

$$BBAND^{+}(p_t, n) = MA(p_t, n) + 2 \times MSD(p_t, n) \tag{5}$$

$$BBAND^{-}(p_t, n) = MA(p_t, n) - 2 \times MSD(p_t, n) \tag{6}$$

- **Momentum (MOM):** MOM measures CPU utilization differences over relatively short periods to follow the speed of the changes in utilization. We used log momentum to center the values at zero. It

is often used to predict reversals [9]. We calculate using (7) as,

$$MOM(p_t, n) = \log(p_t - p_{t-n}) \tag{7}$$

In summary, the selected TIs have been plotted in Fig. 2 after being applied to the PlanetLab dataset (200-time slots), which is described in Section "Dataset".

Then, we study applying and testing Fourier and wavelet transforms as additional features, where Fourier and wavelet transforms are used to remove redundant data and retain the most relevant information [8]. Therefore, these approximation tools could help the deep learning network for predicting trends more accurately.

VTGAN models

We use the GAN network to predict the value and trend of future CPU utilization, i.e., to predict future samples of the time series corresponding to the CPU utilization. Figure 3 illustrates the essential components of the proposed VTGAN architecture. The generator produces CPU traces, which have a *similar* distribution compared to the original CPU traces. The discriminator, however, is responsible for classifying the input trace into either an actual CPU utilization trace or a predicted trace (i.e., an artificially generated CPU utilization trace). The generator and discriminator losses are added together and fed back to the generator to become better at generating CPU utilization traces that mimic the actual data statistics. This process continues until the discriminator no longer be able to differentiate between actual predicted data from generated CPU utilization data.

Some researchers recently reconstructed the generator and the discriminator based on LSTM and CNN layers for better learning regarding several applications. GAN differs from other deep learning techniques in that it tries to strike a balance between the two sides (generator and discriminator) [85].

Figure 4 illustrates the proposed system using the GAN model. In this work, we use an RNN as a generator. Specifically, we employ one of the following recurrent neural networks: (i) LSTM or (ii) GRU, for generating CPU traces. As described in Subsection "Deep learning approaches", RNN has the ability to map generated data from the history of the previous inputs, therefore it is suitable for sequential data. For the discriminator, we utilize a multi-layer 1D-CNN. We choose CNN for the discriminator components as it is able to extract temporal features and information for series data. In the numerical result section, we compare the performance of the two RNNs and select the better generator network.

Regression and classification approaches

Generally, the main goal of forecasting CPU utilization as a time-series forecasting problem is to estimate the closing value of the next time slot. In this work, we focus on CPU utilization value prediction (CPU utilization value regression problem), and the trend direction of CPU utilization (CPU utilization trend classification problem).

A preliminary process, mandatory to follow this approach, is to build a dataset suited to a classification problem. Next, we associate each past observation from the time series with a symbolic label describing the predicted trend (i.e., we label the trend as an upward or a downward trend).

Consequently, we split the dataset into sub-sequences using the *sliding window* technique as input for our models. This technique selects every n samples as inputs, and the $(n+1)$th samples as outputs for value regression and symbolic labels as outputs for trend classification in one-step prediction.

Value regression approach

In this approach, we only focus on predicting the value of CPU utilization and not its trend direction. The CPU utilization value prediction problem has been the traditional approach for proactive resource management in cloud data centers [85]. We use the *sliding window* technique. In this technique, we use the last n samples as an input to our regression technique, i.e., the VTGAN model, to predict future samples. We consider two versions of our scheme, namely, one-step-ahead prediction and p-step-ahead prediction. In the one-step-ahead version, the regression procedure aims to predict the immediate future sample (i.e., one sample only as an output). This is in contrast to the p-step-ahead version, where the regression procedure outputs p future samples.

More specifically, let the input I_{reg} be the CPU utilization time-series samples. The kth row of I_{reg} contains n actual data points (actual CPU utilization), namely, $\{i_k, i_{k+1}, \cdots, i_{n+k-1}\}$, where $k = 1, 2, \cdots, l-n$. We denote the corresponding output by O_{reg}. The output O_{reg} corresponds to the predicted value(s). The kth row of O_{reg} is the predicted CPU utilization at the $(n+k)$th time slot $\hat{i}_{n+k}$ for one-step-ahead prediction, while it is the predicted values $\{\hat{i}_{n+k}, \hat{i}_{n+k+1}, \cdots, \hat{i}_{n+k+p}\}$, as shown in Eqs. (8) and (9) for one-step-ahead and p-step-ahead prediction, respectively.

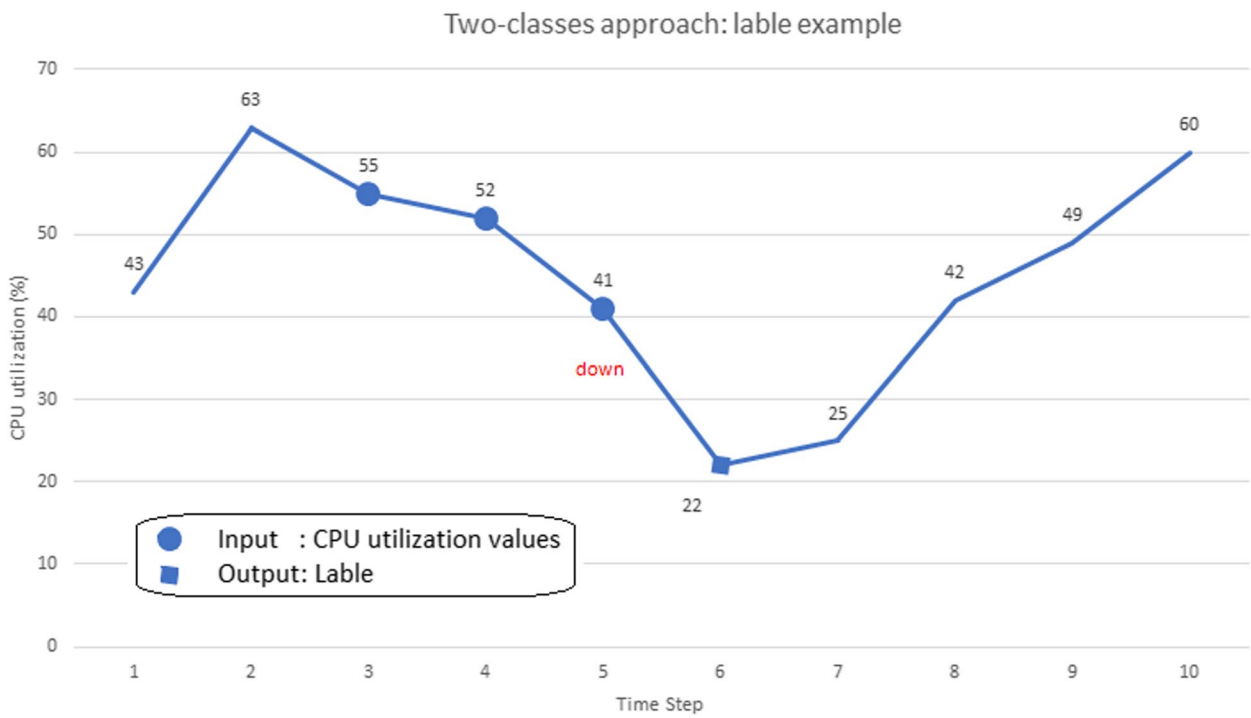


Fig. 5 Assigning label example for trend classification approach

$$I_{reg} = \begin{pmatrix} i_1 & i_2 & \dots & i_n \\ i_2 & i_3 & \dots & i_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ i_r & i_{r+1} & \dots & i_{n+r-1} \end{pmatrix}, \quad O_{reg} = \begin{pmatrix} \hat{i}_{n+1} \\ \hat{i}_{n+2} \\ \vdots \\ \hat{i}_{n+r} \end{pmatrix}, r = l - n + 1 \tag{8}$$

$$I_{reg} = \begin{pmatrix} i_1 & i_2 & \dots & i_n \\ i_2 & i_3 & \dots & i_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ i_r & i_{r+1} & \dots & i_{n+r-1} \end{pmatrix}, \quad O_{reg} = \begin{pmatrix} \hat{i}_{n+1} & \hat{i}_{n+2} & \dots & \hat{i}_{n+p} \\ \hat{i}_{n+2} & \hat{i}_{n+3} & \dots & \hat{i}_{n+p+1} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{i}_{n+r} & \hat{i}_{n+r+1} & \dots & \hat{i}_{n+r+p-1} \end{pmatrix}, r = l - n - p + 1 \tag{9}$$

where i_j denotes the actual CPU utilization at time slot j, $\hat{i}_j$ denotes the predicted CPU utilization at time slot j, n is the sliding window length, and l is the input sequence length.

Trend classification: 2-classes approach

In this section, we describe our proposed algorithm for forecasting the trend of CPU utilization. In this case, we *classify* the direction of the change of the future CPU utilization, whether it is upward or downward. The upward trend of CPU utilization implies that we predict the future CPU utilization to be higher than the current CPU utilization. The downward trend, however, entails that the future CPU utilization is lower than the current CPU utilization. In many practical applications, it is more important to know the trend of workload value rather than the actual value (e.g., in Stock prediction).

Specifically, this approach predicts the *CPU utilization trend* based on two classes:(i) upward and (ii) downward. The movement of each time slot is associated with a label in the set $L = \{up, down\}$, which is determined by comparing the current CPU utilization

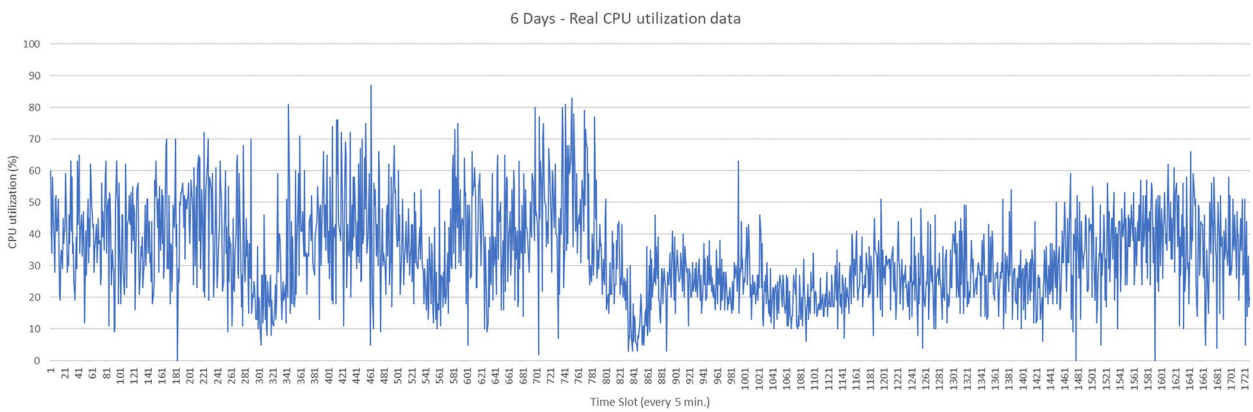


Fig. 6 6 Days CPU utilization data

value to one of the previous time slots. We obtain the class L_m at the mth time slot as follows:

Upward class:

$$\hat{i}_m - i_{m-1} > 0 \Rightarrow L_m = up \tag{10}$$

Downward class:

$$\hat{i}_m - i_{m-1} < 0 \Rightarrow L_m = down \tag{11}$$

where i_{m-1} is the sample of a time series representing the actual value of the CPU utilization at the $(m-1)$th time slot, and $\hat{i}_m$ is the predicted future sample at the mth time slot.

Similar to the CPU utilization value prediction problem, in this approach, we use the sliding window technique in the training procedure to predict the next output trend. We perform the trend prediction in either one-step-ahead prediction fashion or p-step-ahead prediction. The trend prediction of the kth time slot can

Table 2 Selected regression evaluation metrics, their formulas, and symbols

Performance Metric	Equation
Root Mean Squared Error (RMSE)	$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(i_t - \hat{i}_t)^2}$
Mean Absolute Percentage Error (MAPE)	$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\frac{\lvert i_t - \hat{i}_t \rvert}{i_t} \times 100\%$
Mean Absolute Error (MAE)	$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T} \mid i_t - \hat{i}_t \mid$
Theil's coefficient (Theil) [80]	$\text{Theil} = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(i_t - \hat{i}_t)^2}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(i_t)^2} + \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{i}_t)^2}}$
Average relative variance (ARV) [11]	$\text{ARV} = \frac{\sum_{t=1}^{T}(i_t - \hat{i}_t)^2}{\sum_{t=1}^{T}(\hat{i}_t - \bar{i})}$
[11] Prediction of Change in Direction (POCID) [11]	$\text{POCID} = \frac{\sum_{t=1}^{T} D_t}{T} \times 100$ where $D_t = \begin{cases} 1, & \text{if } (i_t - i_{t-1})(\hat{i}_t - \hat{i}_{t-1}) > 0, \\ 0, & otherwise. \end{cases}$
Coefficient of determination (R^2) [11]	$R^2 = 1 - \frac{\sum_{t=1}^{T}(i_t - \hat{i}_t)^2}{\sum_{t=1}^{T}(i_t - \bar{i})^2}$

Symbols:

- T: Number of samples in the time series.
- i_t: True value at time slot t.
- $\hat{i}_t$: Predicted value at time slot t.
- $\bar{i}$: mean value of i

be calculated based on W past observations of the CPU utilization values. We obtain this prediction using the so-called *embedding technique*, i.e., numeric vector input represents a word, by which the vector I_k of past samples is defined as:

$$I_k = \left(i_{k-W+1} \; i_{k-W} \; \ldots \; i_{k-1} \; i_k \right) \tag{12}$$

where W denotes the window size, i.e., the number of data points used to obtain a prediction.

The trend classifier aims at finding a function $f(\cdot)$ that maps the CPU utilization vector I_k into a binary decision $L_{k+1} = \{up, down\}$, i.e., $L_{k+1} = f(I_k)$, where L_{k+1} denotes the predicted trend label at the $(k+1)$th time slot. As CPU utilization time series usually have complex behavior, we propose to employ the VTGAN as a classifier (i.e., for identifying upward or downward trends). Consequently, we capture the non-linear and non-stationary behavior of time series by learning the ML model parameters using data-driven techniques. The input I_{class} is the CPU utilization time-series samples. Each row of I_{class} corresponds to a window of W samples. We organize the samples in a sliding window fashion as in the regression model. The corresponding output O_{class} represents the predicted class value(s), as shown in Eqs. (13) and (14) for one-step-ahead and p-step-ahead prediction, respectively.

$$\begin{aligned} I_{class} &= \begin{pmatrix} i_1 & i_2 & \ldots & i_W \\ i_2 & i_3 & \ldots & i_{W+1} \\ \vdots & \vdots & \ddots & \vdots \\ i_r & i_{r+1} & \ldots & i_{W+r-1} \end{pmatrix}, \\ O_{class} &= \begin{pmatrix} L_{W+1} \\ L_{W+2} \\ \vdots \\ L_{W+r} \end{pmatrix}, r = l - W + 1 \end{aligned} \tag{13}$$

$$\begin{aligned} I_{class} &= \begin{pmatrix} i_1 & i_2 & \ldots & i_W \\ i_2 & i_3 & \ldots & i_{W+1} \\ \vdots & \vdots & \ddots & \vdots \\ i_r & i_{r+1} & \ldots & i_{W+r-1} \end{pmatrix}, \\ O_{class} &= \begin{pmatrix} L_{W+1} & L_{W+2} & \ldots & L_{W+p} \\ L_{W+2} & L_{W+3} & \ldots & L_{W+p+1} \\ \vdots & \vdots & \ddots & \vdots \\ L_{W+r} & L_{W+r+1} & \ldots & L_{W+r+p-1} \end{pmatrix} \\ &\qquad , r = l - W - p + 1 \end{aligned} \tag{14}$$

For instance, Fig. 5 illustrates a label association example using three-sample-window (W=3). The embedded vector at the 5th time slot is as follows:

Table 3 Selected classification evaluation metrics and their formulas

Perfomance metric	Equation
Precision	$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
Recall	$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$
F_1 score (F_1)	$F_1 = 2\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

$$I_5 = \left(55 \;\; 52 \;\; 41 \right) \tag{15}$$

The relative variation from time slot 5 to time slot 6 is:

$$22 - 41 = -19 < 0, \tag{16}$$

and so, the trend label of time slot 6 is $L_6 = down$.

Experimental configuration and evaluation methodology

This section considers the experimental setting used for assessing our proposed prediction models. Our evaluation includes one-step-ahead and p-step-ahead results. We focus our prediction steps p to be limited to 5 (specifically, we focus on $p = 1, 3, 5$ prediction steps). For $p > 5$, we note that the prediction accuracy diminishes. Hence, the prediction outcomes would be less beneficial in practical applications. We compare the accuracy of our proposed VTGAN models against ARIMA, SVR, LSTM, and GRU benchmarks, which appeared in the most recent related works.

Dataset

In our experimental study, we used the PlanetLab traces [60]. These traces contain CPU utilization collected every five minutes from more than 500 places around the world [4]. We show a visual representation of the behavior in Fig. 6, where six days are considered. In particular, CPU utilization values are inputs to predict the value and label for the next time slot. We consider 80% of workload data during all experiments for training the model to predict the remaining data.

Performance evaluation metrics

We investigate various accuracy metrics used to evaluate the proposed VTGAN algorithm. Regarding the CPU utilization value prediction problem, we study the RMSE, MAPE, Theil's coefficient, ARV, POCID, and R^2 coefficient as prediction accuracy (equivalently, evaluate the error in the prediction) metrics. We summarize the formal definitions of the aforestated metrics in Table 2. In the CPU utilization trend classification problem, we consider the precision, the recall, and the

Table 4 The structure of VTGAN models

Model	Layers	Configuration
Stacked LSTM	Bidirectional cuDNNLSTM	256 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNLSTM	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNLSTM	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	FC	Dense, output units (1 or *p* for one or multiple-step-ahead)
		L1 kernel, and bias regularization = 0.00001
Stacked GRU	Bidirectional cuDNNGRU	256 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNGRU	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNGRU	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	FC	Dense, output units (1 or *p* for one or multiple-step-ahead)
		L1 kernel, and bias regularization = 0.00001
VTGAN (LSTM-based)	- Generator parameters	
	Bidirectional cuDNNLSTM	256 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNLSTM	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNLSTM	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	FC	Dense, output units (1 or *p* for one or multiple-step-ahead)
		L1 kernel, and bias regularization = 0.00001
	- Discriminator parameters	
	Conv1D	flter=64, kernel size=5, strides=2, padding=same
		LeakyReLU activation (alpha=0.001)
	Conv1D	flter=128, kernel size=5, strides=2, padding=same
		LeakyReLU activation (alpha=0.001)
	Conv1D	flter=128, kernel size=5, strides=2, padding=same
		LeakyReLU activation (alpha=0.001)
	Flatten	
	FC 1	Dense, units=64, LeakyReLU activation
	FC 2	Dense, output units (1 or *p* for one or multiple-step-ahead),
		sigmoid activation
VTGAN (GRU-based)	- Generator parameters	
	Bidirectional cuDNNGRU	256 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNGRU	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	cuDNNGRU	128 units, dropout= 0.2
		L1 kernel, recurrent, and bias regularization = 0.00001
	FC	Dense, output units (1 or *p* for one or multiple-step-ahead)
		L1 kernel, and bias regularization = 0.00001
	- Discriminator parameters:	
	as VTGAN (LSTM-based)	

Table 5 Comparison of regression results

Model	Window size	Training epochs	Time (Sec.)	RMS E	MAPE	MAE	Theil	ARV	POCID	R^2
ARIMA	5		21.1	8.434	28.175	6.548	0.882	0.867	77.177	0.544
SVR	5		18.9	1.43	4.51	1.12	0.353	0.022	**89.493**	0.981
VTGAN (LSTM-based)	3	3000	74.4	**0.569**±0.16	**1.401**±0.364	**0.464**±0.126	**0.089**±0.078	**0.008**±0.004	88.726±1.62	**0.992**±0.005
CNN-LSTM	10	576	74.8	1.043±0.042	2.407±0.091	0.818±0.042	0.364±0.051	0.028±0.004	79.941±2.622	0.975±0.002
Stacked LSTM	20	626	98.7	1.287±0.006	3.003±0.01	1.018±0.006	0.897±0.016	0.046±0.002	84.768±0.453	0.962±0.0003
VTGAN (GRU-based)	3	3000	68.7	0.755±0.118	1.868±0.279	0.611±0.094	0.125±0.068	0.014±0.004	86.961±0.189	0.987±0.004
CNN-GRU	15	244	34.1	0.94±0.033	2.187±0.094	0.747±0.025	0.282±0.023	0.023±0.001	82.446±2.685	0.979±0.001
Stacked GRU	20	394	65	0.934±0.021	2.145±0.043	0.761±0.016	0.269±0.009	0.024±0.001	88.032±0.747	0.980±0.001

Table 6 Comparison of classification results

Model	Window size	Training epochs	Time (Sec.)	Precision	Recall	F_1 *score*
ARIMA	3		35.2	0.908	0.8681	0.8876
SVR	3		18.7	0.9339	0.895	0.914
VTGAN (LSTM-based)	3	3000	74.4	**0.966**±0.003	**0.900**±0.003	**0.932**±0.002
CNN-LSTM	15	596	80.7	0.929±0.007	0.890±0.024	0.909±0.015
Stacked LSTM	5	364	55.2	0.881±0.008	0.864	0.873±0.004
VTGAN (GRU-based)	3	3000	68.7	0.954±0.009	0.893±0.006	0.922±0.007
CNN-GRU	10	232	35.6	0.915±0.02	0.853±0.01	0.883±0.014
Stacked GRU	5	357	52.2	0.947±0.0002	0.900±0.003	0.923±0.002

F_1 score as classification accuracy metrics. We summarize the formal definitions of the classification accuracy metrics in Table 3. In addition, we use the confusion matrix as a visual evaluation to reflect the classifier's recognition ability for each class. We show the confusion matrix in terms of a 2-class approach (upward and downward) for the trend classification problem, while we use 10 quantized classes for the regression problem. Specifically, we quantize the CPU utilization percentage into 10 classes (in steps of 10%). Hence, we have classes $0, 1, 2, \cdots, 9$ representing the CPU utilization percentages of $> 90\%, 80 - 90\%, 70 - 80\%, \cdots, 0 - 10\%$.

We select RMSE, MAPE, MAE, and ARV for regression evaluation metrics to measure the deviation between the predicted and actual values. With all these metrics, the absolute value of the error prevents the positive and negative errors from canceling out each other. The MAPE metric, in particular, has the added benefit of allowing prediction accuracy comparison of time series with different value scaling.

Theil's coefficient measures relative accuracy that compares the obtained predicted results with actual values by giving more weight to massive errors by squaring the deviations. Theil coefficient acceptable ranges from 0 (corresponding to no forecasting error) and 1 (corresponding to no predictive ability). More than 1 value means poor prediction guessing [80, 83].

POCID measures the capability of predicting if future values will increase or decrease. It is superior to MAPE as it measures the prediction accuracy based on its change direction. Therefore, it is a powerful metric during the decision-making stage. POCID value closer to 100 represents the best value [11].

R^2 represents the coefficient of how close the values are to be fitted with the line of regression. If R^2 value equal to 1, this means that the model perfectly fits all variability. Therefore, R^2 value closer to 1 represents the best value [11].

For the classification problem, we evaluate the accuracy of the proposed model using the precision, the recall, and the F_1 score.

Experiment configuration

We perform all experiments on Intel Xeon Gold 6248 processor with 2.5 GHz clock speed, 128 GB of memory, and a Tesla V100 GPU with 32 GB of RAM. We implement all deep learning models using the Keras framework and Tensorflow backend with CuDNN kernels. Table 4 illustrates the architecture of proposed models.

We set the batch size and epochs to 32 and 3000, respectively, regarding the training phase. For hybrid CNN-LSTM/CNN-GRU and stacked LSTM/GRU models, the early stopping technique is used with a 20% validation rate. This technique finds the best point to halt the optimizer (Root Mean Squared Propagation - RMSprop) once the model performance stops improving [53]. We configure the stacked LSTM/GRU network structures as the generator configurations of VTGAN models. Also, the loss function for the generator is the mean squared error after the try-and-error method. We test each model three times, then the average and the standard deviation are calculated.

Results and discussions

This section presents the regression and classification accuracy results of the proposed VTGAN models. Subsections "One-step-ahead regression and classification accuracy results", "Regression and classification accuracy results using technical indicators", and "Multistep-ahead regression and classification accuracy results for different sliding window size" show the experimental results of the proposed algorithm compared to traditional models in recent publications such as CNN-LSTM/CNN-GRU and stacked-LSTM/GRU. Also, Section "Bitbrains dataset comparison" illustrates an additional evaluation study with another real cloud dataset (Bitbrains).

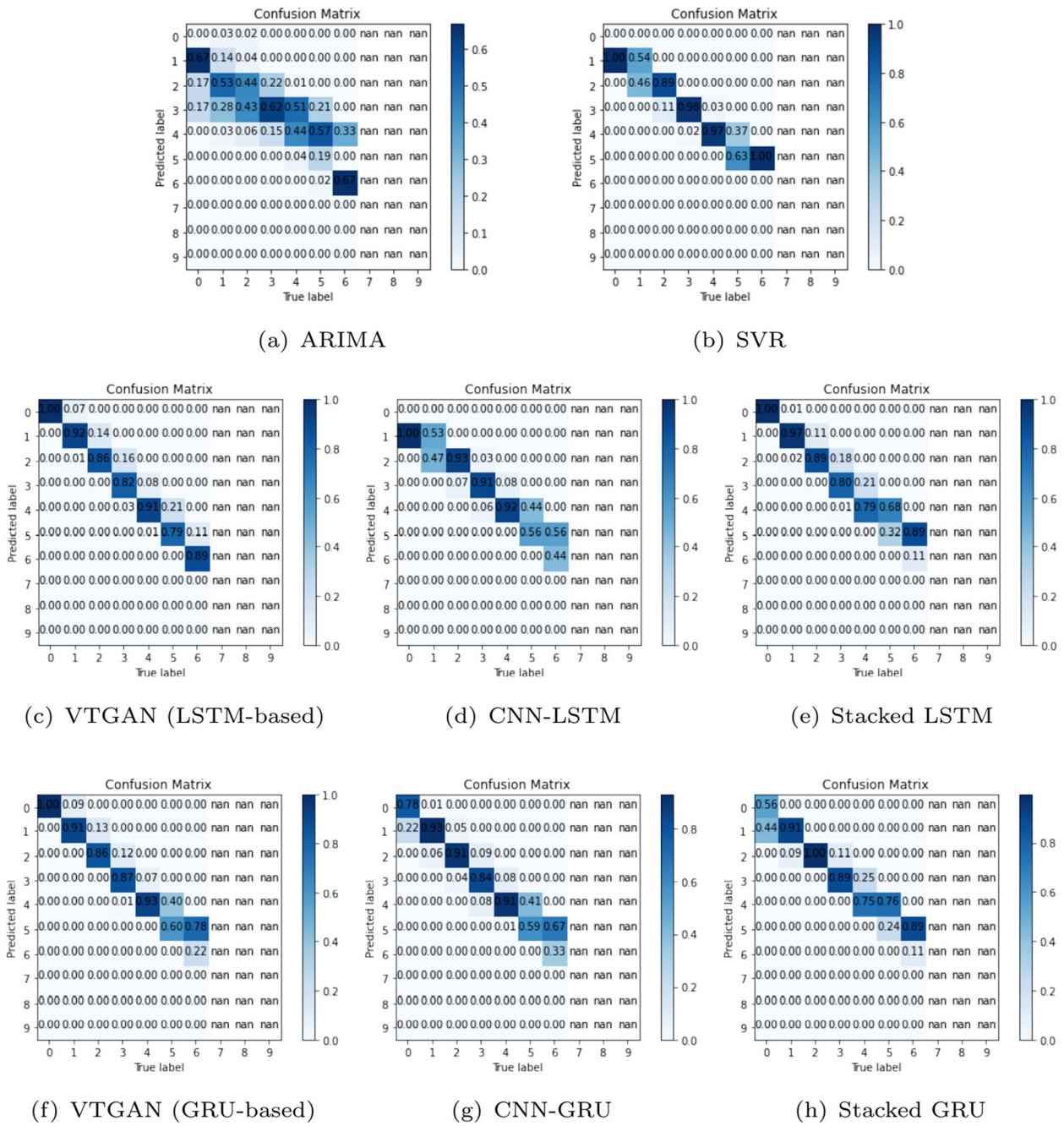


Fig. 7 Confusion matrices for regression approach. Classes $0, 1, 2, \cdots, 9$ represent the CPU utilization percentages of $> 90\%, 80 - 90\%, 70 - 80\%, \cdots, 0 - 10\%$, respectively

One-step-ahead regression and classification accuracy results

In this section, we assess the performance of VTGAN models in one-step-ahead regression and classification approaches. We optimize the window size such that it achieves maximum accuracy. Tables 5 and 6 illustrate the overall accuracy performance of VTGAN models compared to other models for regression and classification approaches, respectively. In addition, These tables show the optimal values for window size, stopped training

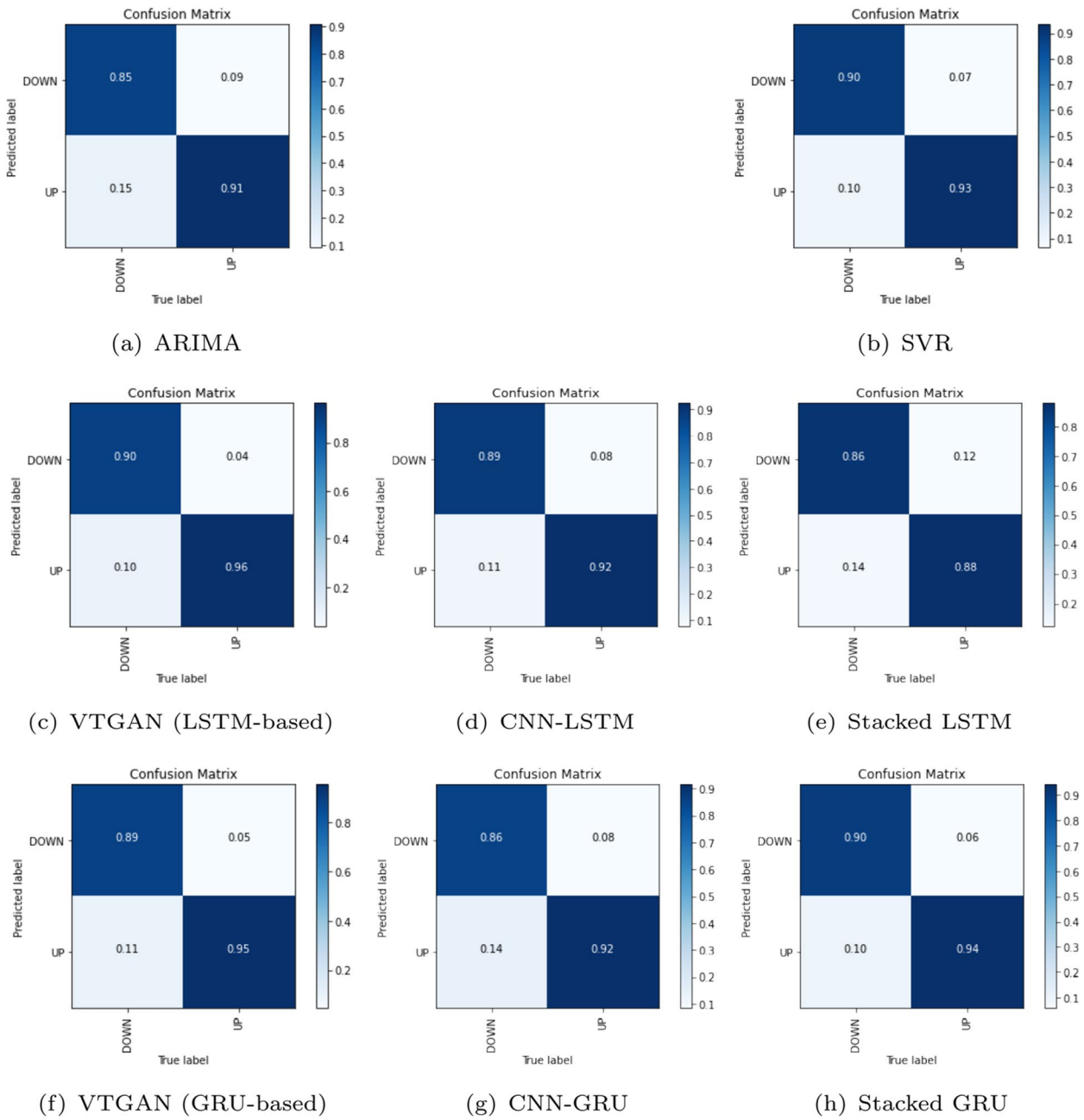
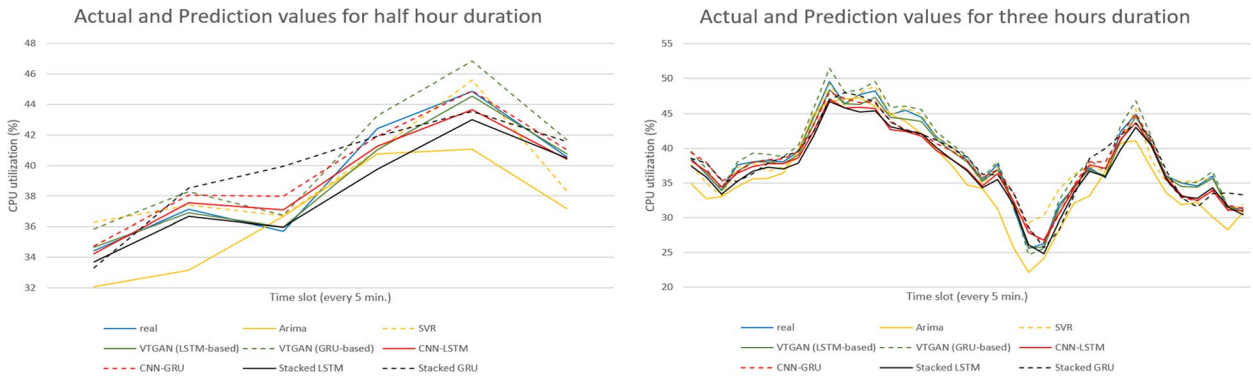


Fig. 8 Confusion matrices for classification approach

epochs, and training time for the best-observed performance in each model. In all tables, the best-observed model is in bold in each approach.

As we can see from the experimental results, VTGAN (LSTM-based) model is superior to all other prediction models, whether for regression or classification approaches regarding all performance metrics presented in Section "Performance evaluation metrics". The stacked LSTM model performs the worst compared with all DL techniques. Although, the results of the stacked LSTM remain acceptable since Theil value does not exceed one. Although the SVR model achieves a higher POCID value, it did not exceed the maximum value of VTGAN (LSTM-based) after adding the standard deviation.

Focusing on the sliding window size (from the Tables 5 and 6, $W = 3$, which is equivalent to 15 minutes), VTGAN models achieve higher performance with small sliding window sizes, whether using LSTM or GRU as a



(a) Half hour (b) Three hours

Fig. 9 Actual and predicted CPU utilization values for regression approach

generator. This result agrees that the small window size is more suitable for the drift data as cloud workload data, while larger window sizes are more appropriate for noisy data [78]. Nevertheless, since LSTM and GRU techniques capture long-term dependencies [19, 42], the regression and classification accuracy of LSTM/GRU models enhances with a longer window size value relative to VTGAN models.

Hybrid and deep learning-based models are usually more complex and require higher computations for model training. Nevertheless, for all tested models, the training time is acceptable for resource management applications of the data center because overload/underload detection processes often occur every 5 minutes as in [12, 33]. As shown in Tables 5 and 6, the CNN-GRU model achieves less training time and epochs number whether regression or classification approaches (see underlined values in Tables 5 and 6).

We note that the complexity difference between models is a consequence of using the early stopping technique. Also, Tables 5 and 6 show that GRU-based models record less training time and the number of epochs compared to the LSTM-based models. This observation is consistent with the fact that the GRU-based models are much less computationally intensive. This is due to their ability to converge with fewer parameters [20]. However, the performance accuracy of the VTGAN (LSTM-based) model is superior to the VTGAN (GRU-based) model for all tested models.

Figures 7 and 8 illustrate the confusion matrices of all models. We use the confusion matrix comparison to visually examine the behavior of VTGAN models compared to others with regression and classification results, respectively. Also, Fig. 9 illustrates a part of the actual CPU utilization compared to the predicted value using all models. The interval length is of 5 minutes.

The confusion matrix results of regression models in Fig. 7 illustrate the predictive capability within every CPU utilization interval. Figure 7 shows that the VTGAN (LSTM-based) model is superior in overall prediction accuracy. VTGAN (LSTM-based) model achieves accurate prediction at every CPU utilization range. In contrast, the prediction accuracy reduces for very low or very high CPU utilization values compared to other models, particularly for the ARIMA, SVR, and CNN-LSTM models, as shown in Fig. 9.

The confusion matrix results of classification models in Fig. 8 signify the classification accuracy for predicting upward or downward trends. Figure 8, VTGAN (LSTM-based) model achieves the best performance, followed by VTGAN (GRU-based) and stacked GRU models, which record slightly less accuracy. The strength of the classification approach is that it is easy to make direct decisions depending on the classifier results. For instance, we can detect the overloaded server if its CPU utilization records more than a specific threshold and the predicted trend is upward. This solution will reduce unnecessary migrations in resource management frameworks. Especially, the False downward detection probability with VTGAN (LSTM-based) model is low ($\approx$ 4%).

Regression and classification accuracy results using technical indicators

This section analyzes the impact of adding Technical Indicators (TIs) to the feature set with our workload traces. By repeating previous experiments in Section "One-step-ahead regression and classification accuracy results", Tables 7 and 8 illustrate the overall accuracy performance of VTGAN models using TI strategy compared to other models for regression and classification approaches, respectively.

Table 7 Comparison of regression results

Model	Window size	Training epochs	Time (Sec.)	RMSE	MAPE	MAE	Theil	ARV	POCID	R^2
VTGAN (LSTM-based)	3	3000	74.2	1.256±0.014	3.075±0.077	1.013±0.021	0.383±0.033	0.039±0.001	79.235±0.444	0.963±0.001
CNN-LSTM	15	235	35	1.776±0.013	4.186±0.058	1.404±0.02	1.310±0.13	0.078±0.005	75.247±0.452	0.927±0.001
Stacked LSTM	20	381	62.7	1.449±0.012	3.444±0.045	1.151±0.009	0.699±0.025	0.053±0.002	74.975±0.685	0.951±0.001
VTGAN (GRU-based)	3	3000	72	**1.096**±0.013	**2.669**±0.044	**0.887**±0.015	**0.242**±0.037	**0.029**±0.001	**80.490**±0.74	**0.972**±0.001
CNN-GRU	15	209	29.3	1.685±0.04	3.958±0.221	1.314±0.053	1.213±0.025	0.069±0.007	5.345±0.745	0.934±0.003
Stacked GRU	20	255	48.9	1.492±0.012	3.490±0.011	1.155±0.002	0.788±0.032	0.053±0.001	70.524±0.747	0.948±0.001

Table 8 Comparison of classification results

Model	Window size	Training epochs	Time (Sec.)	Precision	Recall	F_1 *score*
VTGAN (LSTM-based)	3	3000	74.2	0.826±0.002	0.806±0.018	0.816±0.009
CNN-LSTM	15	235	35	0.782±0.004	0.775±0.017	0.778±0.007
Stacked LSTM	15	512	73.2	0.784±0.005	0.771±0.003	0.778±0.001
VTGAN (GRU-based)	3	3000	72	**0.854**±0.002	**0.804**±0.003	**0.828**±0.001
CNN-GRU	20	235	35	0.803±0.016	0.788±0.017	0.795±0.01
Stacked GRU	20	255	48.9	0.743±0.007	0.718±0.009	0.730±0.008

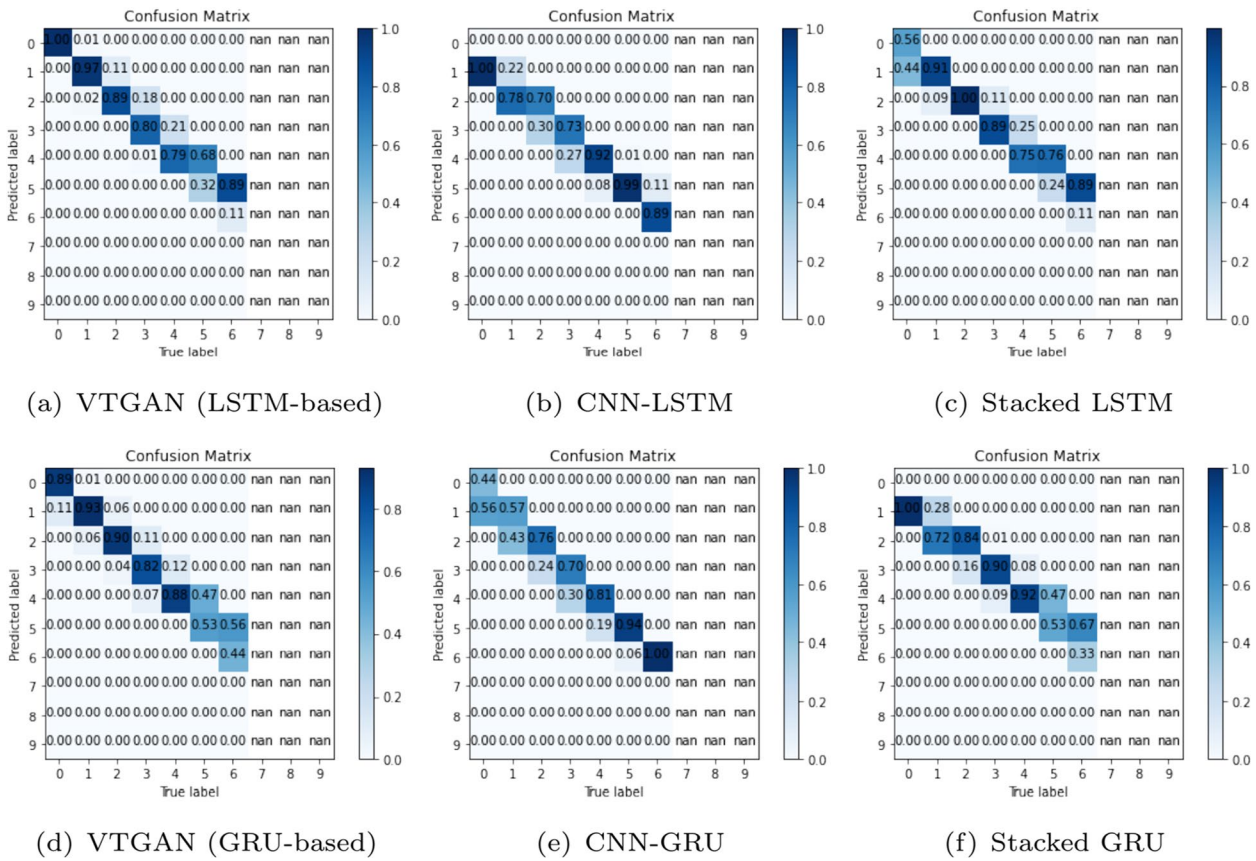


Fig. 10 Confusion matrices for regression approach using TI. Classes $0, 1, 2, \cdots, 9$ represent the CPU utilization percentages of $> 90\%, 80 - 90\%, 70 - 80\%, \cdots, 0 - 10\%$, respectively

In general, the TI addition diminishes the regression and classification performance for all tested models in terms of one-step-ahead prediction. This result could be due to the occurrence of over-fitting by adding dependent features. VTGAN models are still the superior models for regression and classification approaches.

VTGAN (GRU-based) model outperforms other models (bold results). In contrast, CNN-LSTM/GRU models are the worst performance. In this case, the regression becomes useless, where the Theil value of these models record exceeds one, as shown in Table 7.

Figures 10 and 11 illustrate the comparison of confusion matrices between all the models using TIs strategy to examine the visual behavior of VTGAN models compared to others.

Focusing on the training speed of the models, we note that the single benefit of using the TI strategy for one-step-ahead prediction is that the training is

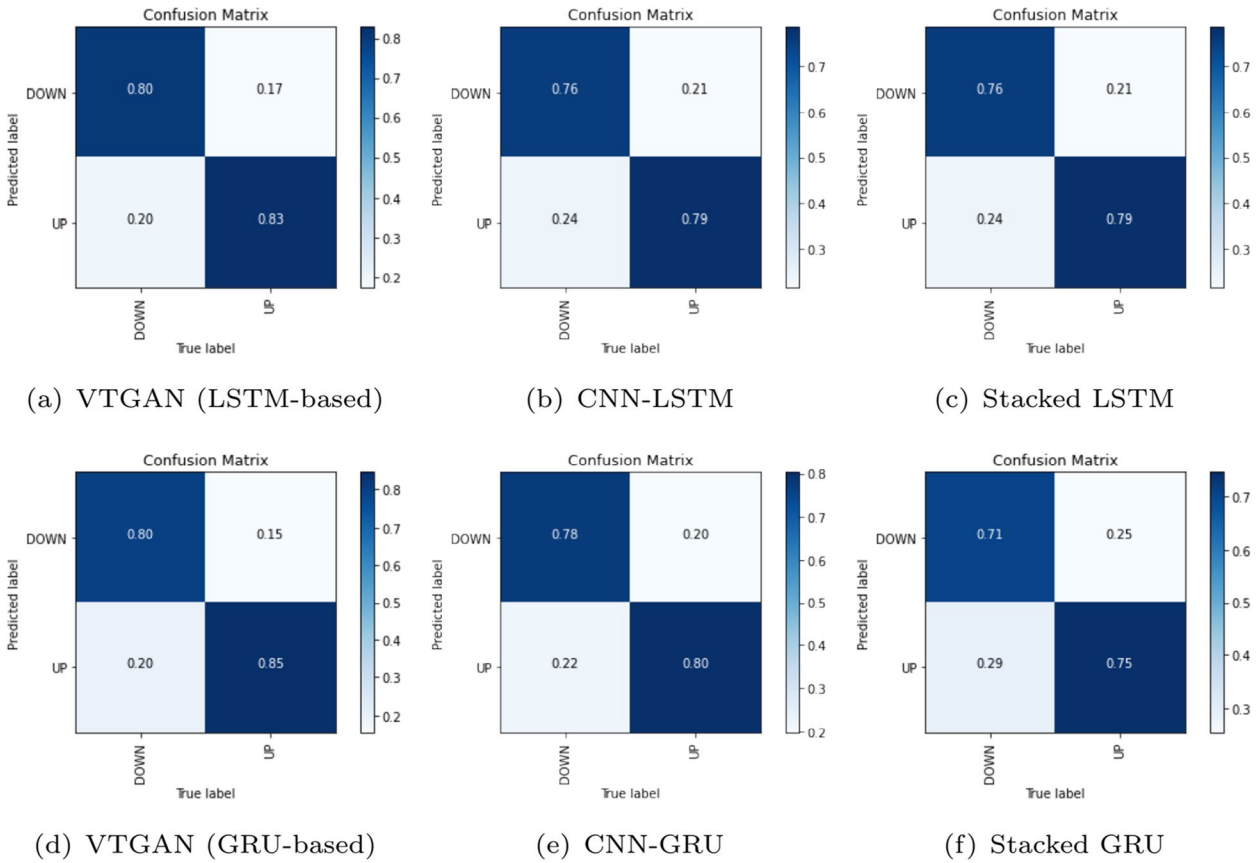


Fig. 11 Confusion matrices for classification approach using TI

faster than others. Specifically, the training time and the number of epochs reduce for CNN-LSTM/GRU and stacked LSTM/GRU models, whether regression or classification approaches compared to the results in Subsection "One-step-ahead regression and classification accuracy results". For instance, the training epochs and time decrease from 576 and 74.8 seconds in Table 5 to 235 and 35 seconds using the TI strategy for the CNN-LSTM model in Table 7.

Multistep-ahead regression and classification accuracy results for different sliding window size

This section studies the performance of the multi-step-ahead prediction. Also, we assess the effect of changing the sliding window sizes on our models' performance and/or adding TI features to the input of the prediction algorithm. The following subsections analyze the impact of change in sliding window size, multi-step-ahead, and TI strategy, respectively.

Sliding window size analysis

This section analyzes the effect of changing the sliding window size. Figures 12 and 13 illustrate MAPE and F_1 score values against the sliding-window size for all tested models. Sub-figures in every row represent the step-ahead size ($p = 1, 3, 5$). The second column represents the results after adding the TI indicators.

Figures 12 and 13 show that VTGAN models' performance significantly declines when the sliding window size increases. In contrast, the performance of other models oscillates to a reasonable degree. Fortunately, the VTGAN models' accuracy outperforms other models with small window sizes. This result is considered a considerable benefit when we run our model for real-time resource management framework as in [33]. This result implies that as soon as the model collects *three* CPU utilization data points (i.e., in a period of 15 minutes), it can successfully predict future samples.

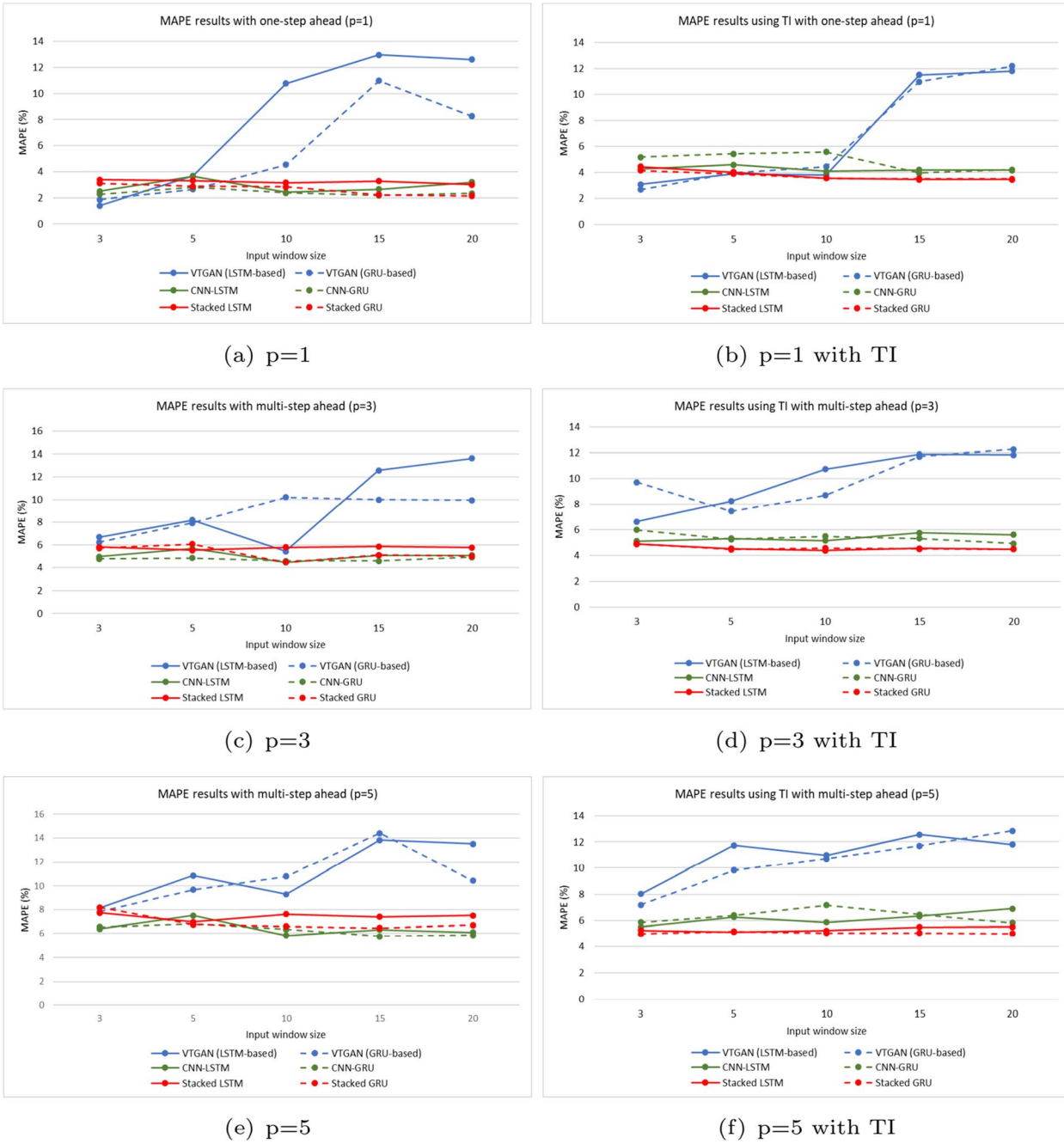


Fig. 12 MAPE values using different window size input for one-step and mult-step ahead

Technical indicators effect on multi-step-ahead prediction

This section analyzes the impact of using TIs for all tested scenarios with different sliding windows and step-ahead sizes. Figures 14 and 15 illustrate MAPE and F_1 score values, respectively. Solid and striped bars represent the pure models and models using the TIs, respectively, with various sliding window sizes (3, 5, 10, 15, and 20) and step-ahead sizes ($p = 1, 3, 5$).

In general, the performance of all models with multi-step-ahead fails to maintain its performance whenever the prediction step size increases for all tested configurations. As shown in solid bars only in Figs. 14 and 15.

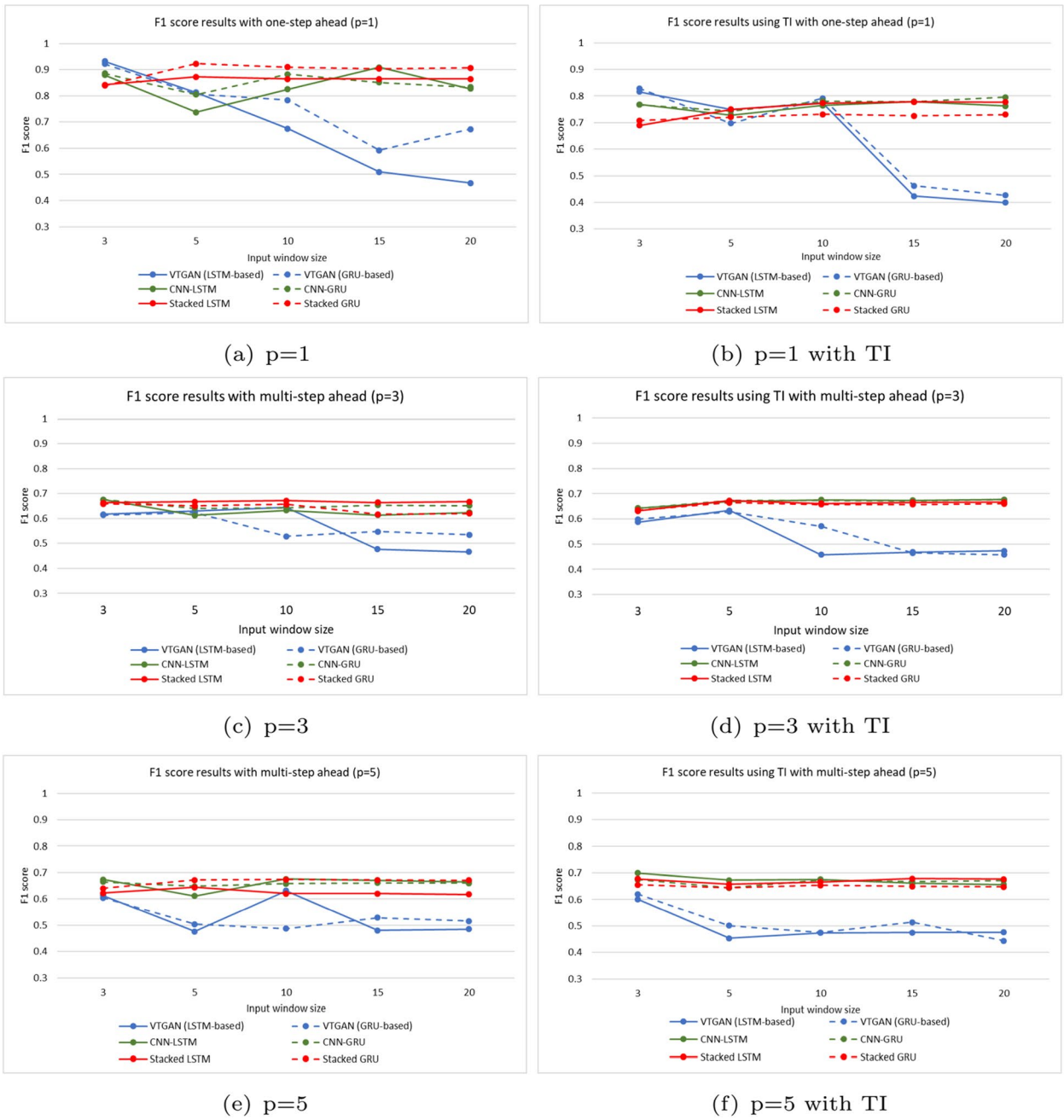


Fig. 13 F_1 score values using different window size input for one-step and multi-step ahead

This result agrees with the results in [61, 85], which confirmed that most deep-learning and hybrid models perform poorly in long-term prediction approaches. That is because of the nature of CPU utilization data, where it fails to fit models due to the complexity and non-linearity issues.

Regarding the one-step-ahead prediction, the use of the TI strategy negatively affects the regression and classification performance except for the VTGAN (LSTM-based) model. It achieves a significant improvement for window size equals 10 (Figs. 14(g) and 15(g)), then a slight improvement in regression performance for window sizes equal 15 and 20 (Fig. 14(j) and (m)).

Regarding multi-step-ahead regression, the use of the TI strategy achieves a significant improvement with stacked LSTM/GRU models (Fig. 14(columns 2 and 3)).

(a) Window size=3, p=1 (b) Window size=3, p=3 (c) Window size=3, p=5

(d) Window size=5, p=1 (e) Window size=5, p=3 (f) Window size=5, p=5

(g) Window size=10, p=1 (h) Window size=10, p=3 (i) Window size=10, p=5

(j) Window size=15, p=1 (k) Window size=15, p=3 (l) Window size=15, p=5

(m) Window size=20, p=1 (n) Window size=20, p=3 (o) Window size=20, p=5

Fig. 14 MAPE values using different window size and step ahead values

(a) Window size=3, p=1 (b) Window size=3, p=3 (c) Window size=3, p=5

(d) Window size=5, p=1 (e) Window size=5, p=3 (f) Window size=5, p=5

(g) Window size=10, p=1 (h) Window size=10, p=3 (i) Window size=10, p=5

(j) Window size=15, p=1 (k) Window size=15, p=3 (l) Window size=15, p=5

(m) Window size=20, p=1 (n) Window size=20, p=3 (o) Window size=20, p=5

Fig. 15 F_1 score values using different window size and step ahead values

Table 9 Best configuration based on the number of step-ahead prediction sizes

Prediction sizes	Value regression	Trend classification
1-step-ahead	VTGAN (LSTM-based)	VTGAN (LSTM-based)
	Window size=3	Window size=3
	(Fig. 14(a))	(Fig. 15(a))
3-step-ahead	Stacked LSTM	CNN-LSTM
	Window size=10	Window size=20
	With TIs	With TIs
	(Fig. 14(h))	(Fig. 15(n))
5-step-ahead	Stacked GRU	CNN-LSTM
	Window size=3	Window size=3
	With TIs	With TIs
	(Fig. 14(c))	(Fig. 15(c))

Regarding multi-step-ahead classification, the use of the TI strategy achieves a slight improvement with the stacked LSTM model and most CNN-LSTM/GRU models (Fig. 15(columns 2 and 3)).

Table 9 illustrates the best configurations based on the number of prediction steps for regression and classification approaches. Service providers can choose the model and adjust the configuration based on the required prediction steps. For one-step-ahead prediction, VTGAN (LSTM-based) model outperforms other models with a window size equal to 3 (15 minutes), whether regression or classification approaches. For multi-step-ahead prediction, Stacked LSTM/GRU and CNN-LSTM outperform other models with TIs for the regression and classification approach, respectively.

In general, the use of the TI strategy is powerful in the case of long-term prediction strategy in some models. Unfortunately, this is not suitable for real-time resource management frameworks in cloud data centers, and that might be because adding dependent features leads to an over-fitting issue. Nevertheless, this issue is promising to investigate and could be improved using ensemble and hybrid strategies as in [86].

Bitbrains dataset comparison

To confirm the performance evaluation of the proposed models, we perform experiments using another real cloud dataset, namely, Bitbrains [72]. This dataset is published online in the Grid workloads archive [10]. It is a large-scale and long-term trace of real data. The dataset of Bitbrains contains data spanning over 5,446,811 CPU hours (1750 VMs), with 23,214 GB memory and 5,501 cores. For comparison purposes, we perform the same preprocessing steps as [44]. Then, we evaluate our proposed models compared to the models of Authors in [44] with the regression approach only, as using the trend

Table 10 Prediction performance of Bitbrains dataset for the proposed models compared to the models in [44]

Method	Window size	Train:Test ratio	MAPE
Bi-LSTM [44]		65:35	12.0119
		70:30	12.2173
	30	75:25	12.3019
		80:20	13.6177
		65:35	**11.7046**
		70:30	11.7091
	60	75:25	11.914
		80:20	13.6163
		65:35	12.0244
		70:30	12.3091
	90	75:25	12.8671
		80:20	13.1198
		65:35	12.0802
		70:30	11.8903
	120	75:25	14.207
		80:20	13.4428
BHyPreC [44]		65:35	11.1799
		70:30	12.3343
	30	75:25	12.3688
		80:20	12.2959
		65:35	11.1101
		70:30	13.0751
	60	75:25	11.7641
		80:20	13.507
		65:35	12.537
		70:30	12.2912
	90	75:25	10.8557
		80:20	12.4713
		65:35	12.2044
		70:30	**10.7738**
	120	75:25	12.706
		80:20	13.3193
VTGAN (LSTM-based)		65:35	10.5822
		70:30	9.47898
	30	75:25	9.39637
		80:20	**9.0233**
		65:35	10.911
		70:30	10.1507
	60	75:25	10.4705
		80:20	9.3998
		65:35	10.3466
		70:30	13.6877
	90	75:25	11.146
		80:20	11.2193
		65:35	13.0493
		70:30	14.7279
	120	75:25	12.4581
		80:20	12.8819
VTGAN (GRU-based)		65:35	8.87

Table 10 (continued)

Method	Window size	Train:Test ratio	MAPE
		70:30	8.6018
	30	75:25	9.1799
		80:20	9.6228
		65:35	8.5347
		70:30	8.4522
	60	75:25	9.044
		80:20	**8.1686**
		65:35	8.747
		70:30	8.8152
	90	75:25	8.5942
		80:20	8.3724
		65:35	8.6346
		70:30	9.0875
	120	75:25	8.0545
		80:20	8.5751

Table 11 Summary of lowest MAPE values of our proposed models compared to the models in [44]

Method	Best tested window size	Best tested split ratio	Lower MAPE value
ARIMA [44]	N/A	80:20	37.031
LSTM [44]	120	65:35	11.7246
GRU [44]	90	70:30	11.9765
Bi-LSTM [44]	60	65:35	11.7046
BHyPreC [44]	120	70:30	10.7738
VTGAN (LSTM-based)	30	80:20	9.0233
VTGAN (GRU-based)	60	80:20	**8.1686**

Table 12 MAPE percentage increase/decrease of the compared models in [44] with respect to our proposed model

Compared models	VTGAN (LSTM-based)	VTGAN (GRU-based)
ARIMA	+310.3931%	+319.8652%
LSTM	+29.9369%	+39.4091%
GRU	+32.7286%	+42.2007%
Bi-LSTM	+29.7153%	+39.1874%
BHyPreC	+19.3998%	+28.8719%

classification is a novel approach in the field of cloud workload forecasting.

Table 10 illustrates the MAPE of CPU utilization prediction with the values of the same variables that are used in [44], such as window size and train/test ratio. Also, Table 11 illustrates the lowest MAPE value for each model with optimum window size and split ratio, which is obtained from all combinations shown in Table 10.

We can see that our proposed models achieve the highest prediction accuracy compared to other state-of-the-art prediction models in [44]. However, the lowest MAPE is obtained in our VTGAN (GRU-based) model for a window size of 60 and a split ratio of 80:20. The split size ratio remains the same for our VTGAN (LSTM-based) model, but history window size changes to 30.

Table 12 illustrates the improvement or diminishing percentage of using our proposed models compared to the state-of-the-art prediction models. We calculate it as [44] using the Eq. (17), where Y_p and Y_c denote the MAPE value of our proposed model and the compared model, respectively. We take into consideration the best combination scenario only for all the models in terms of window size and split ratio.

$$X_c = \frac{(Y_c - Y_p) * 100}{Y_p} \tag{17}$$

For this comparison study, we use ARIMA, LSTM, GRU, Bi-LSTM, and BHyPreC as the baseline models to compare. The Positive percentage denotes the percentage increase of the MAPE value of the compared model with respect to our proposed models. We clearly see that the percentage MAPE value increases for all the models compared to our proposed models.

As we can see, our proposed models considerably minimize the MAPE in predicting CPU utilization. Therefore, our models are not only superior to the classical models (ARIMA) but also perform much better compared to other deep learning approaches presented in this paper.

Conclusions and future works

In recent years, the workload prediction process has become a key stage towards efficient resource allocation and management approaches in cloud computing environments. Due to the non-linearity of cloud workloads, this issue faces enormous challenges. Therefore, this paper proposes a novel direction in the cloud workload prediction field by considering the future movement direction in a modern classification structure. In addition, it presents novel VTGAN models, which are based on a GAN network with stacked LSTM or GRU as a generator and 1D CNN as a discriminator. The main benefit of VTGAN models is their ability to deal effectively with long-term nonlinear dependencies of cloud workloads.

In this paper, we study the proposed models on different configurations over an over-volatile real cloud workload trace. Also, we present the impact of tuning sliding

window size and multi-step-ahead strategy. In addition, we study the use of technical indicators, Fourier transforms, and wavelet transforms to increase the number of input features. We apply all of these studies with the VTGAN models compared to stacked LSTM/GRU and CNN-LSTM/GRU models.

The experimental results demonstrate that the VTGAN models are superior for the cloud workload prediction approach, whether using LSTM or GRU as a generator. Also, these results illustrate the effectiveness of transforming the problem to classify the trend instead of predicting the value of future workload for all tested models. Significantly, the upward classification accuracy reaches 96.6%. The proactive overload detection stage in the resource management techniques is a critical issue that overcomes the unnecessary migrations that violate the service level agreement for end-users. The results are not promising regarding the multi-step-ahead prediction and technical indicator strategies. Thus, one-step-ahead prediction is more suitable for a real-time cloud environment. In addition, the technical indicator approach may be extended further by proposing a solution to optimize the prediction and classification error.

As an additional suggestion for future work, a dynamic scaling method can be applied rather than set a fixed value to improve the prediction and classification accuracy. Another future direction is to implement these prediction models in an actual resource management framework for the cloud data center through the CloudSim simulation tool to evaluate the proposed models in a large-scale simulated cloud environment. Hence, the decision of resource allocation will be based on the trend. In addition, we will extend the classification approach so that the CPU utilization trend will be predicted based on three classes:(i) upward trend, (ii) hold, and (ii) downward trend.

As further promising directions for future research, our contribution opens research areas concerning next-generation computing, such as Edge AI [75]. Especially, a hybrid solution could be presented by processing real-time applications on edge devices and training models on the cloud [50, 65]. Our trend classification approach could be helpful in this Edge-to-cloud integration approach in offloading the training process to the cloud by allocating it to the best host, depending on the future workload of the servers. This approach could be considered and implemented for most resource allocation frameworks, such as Mobile edge computing and fog computing platforms for internet of things (IoT) purposes [49]. That approach increases computational performance and reduces the total energy consumed and processing times for mobile or edge devices. Moreover, edge computational resources suffer from QoS degradation due to overloading and inconsistency. Therefore, an intelligent proactive workload management framework could be presented to guarantee the load balancing between the edge resources using our classification approach.

Abbreviations

ANNs	Artificial neural networks
ARMA	Autoregressive moving average
ARMAX	Autoregressive moving average with exogenous inputs
ARV	Average relative variance
BBANDs	Bollinger bands
CMA	Cumulative moving average
CNN	Convolution neural network
DES	Double exponential smoothing
DM	Difference model
DL	Deep learning
DLL	Dynamically linked library
EMA	Exponential moving average
GAN	Generative adversarial network
GRU	Gated recurrent units
ETS	Error trend seasonal exponential smoothing
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
LR	Linear regression
LSTM	Long short-term memory
MAs	Moving averages
MACD	Moving average convergence divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine learning
MLR	Multiple linear regression
MM	Median model
MSD	Moving standard deviation
MOM	Momentum
POCID	Prediction of Change in Direction
QoS	Quality of service
RMSE	Root Mean Squared Error
RNN	Recurrent neural network
SES	Simple exponential smoothing
SVM	Support vector machine
SVR	Support vector regression
TIs	Technical indicators
WMA	Weighted moving average
VM	Virtual machine
VTGAN	Value trend generative adversarial network

Acknowledgements

The workload traces used in this work were provided by Bitbrains IT Services Inc. from the GRID Workloads Archive. Also, the authors would like to acknowledge and be grateful to the anonymous reviewers for their wonderful suggestions.

Authors' contributions

Conceptualization, Aya, Noha, Karim, Hanan and Walaa; methodology, Aya, Noha, Karim, Hanan and Walaa; implementation, Aya; validation, Aya, Noha, Karim, Hanan and Walaa; writing— original draft preparation, Aya, Noha and Karim; writing—review and editing, Aya, Noha, Karim, Hanan and Walaa; supervision, Noha, Karim, Hanan and Walaa. The authors read and approved the final manuscript.

Funding
Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Availability of data and materials
The data required to support these findings cannot be shared at this time as the data also forms part of an ongoing Ph.D. thesis.

Declarations

Consent for publication
Consent has been granted by all authors and there is no conflict.

Competing interests
The authors declare no competing interests.

Received: 7 January 2023 Accepted: 12 June 2023
Published online: 26 June 2023

References

1. Alibaba cluster traces. https://github.com/alibaba/clusterdata. Accessed Mar 2022
2. Dinda. http://www.cs.cmu.edu/~pdinda/LoadTraces/. Accessed Mar 2022
3. Google cluster data. https://github.com/google/cluster-data. Accessed Apr 2022
4. The planetlab traces. http://github.com/beloglazov/planetlab-workload-traces. Accessed May 2022
5. Wikimedia foundation. http://dumps.wikimedia.org/other/pagecounts-raw. Accessed May 2022
6. Ajila SA, Bankole AA (2013) Cloud client prediction models using machine learning techniques. In: 2013 IEEE 37th Annual Computer Software and Applications Conference, IEEE, pp 134–142. https://doi.org/10.1109/COMPSAC.2013.21
7. Aldossary M, Alzamil I, Djemame K (2017) Towards virtual machine energy-aware cost prediction in clouds. In: International Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, pp 119–131. https://doi.org/10.1007/978-3-319-68066-8_10
8. Alegeh N, Thottoli M, Mian N, Longstaff A, Fletcher S (2021) Feature extraction of time-series data using dwt and fft for ballscrew condition monitoring. In: Advances in Manufacturing Technology XXXIV: Proceedings of the 18th International Conference on Manufacturing Research, Incorporating the 35th National Conference on Manufacturing Research, 7-10 September 2021, University of Derby, Derby, UK, IOS Press, vol 15, p 402. https://doi.org/10.3233/ATDE210069
9. Alonso-Monsalve S, Suárez-Cetrulo AL, Cervantes A, Quintana D (2020) Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. Expert Syst Appl 149(113):250. https://doi.org/10.1016/j.eswa.2020.113250
10. Anoep S, Dumitrescu C, Epema D, Iosup A, Jan M, Li H, Wolters L The grid workloads archive: Bitbrains. http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. Accessed June 2022
11. Anuradha J et al (2021) Big data based stock trend prediction using deep CNN with reinforcement-LSTM model. Int J Syst Assur Eng Manag 1–11. https://doi.org/10.1007/s13198-021-01074-2
12. Beloglazov A, Buyya R (2015) Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. Concurr Comput Pract Experience 27(5):1310–1333. https://doi.org/10.1002/cpe.3314
13. Bi J, Li S, Yuan H, Zhou M (2021) Integrated deep learning method for workload and resource prediction in cloud systems. Neurocomputing 424:35–48. https://doi.org/10.1016/j.neucom.2020.11.011
14. Bi J, Li S, Yuan H, Zhao Z, Liu H (2019) Deep neural networks for predicting task time series in cloud computing systems. In: 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), IEEE, pp 86–91. https://doi.org/10.1109/ICNSC.2019.8743188
15. Biswas NK, Banerjee S, Biswas U, Ghosh U (2021) An approach towards development of new linear regression prediction model for reduced energy consumption and sla violation in the domain of green cloud computing. Sustain Energy Technol Assess 45:101087. https://doi.org/10.1016/j.seta.2021.101087
16. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst 25(6):599–616. https://doi.org/10.1016/j.future.2008.12.001
17. Calheiros RN, Masoumi E, Ranjan R, Buyya R (2014) Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Trans Cloud Comput 3(4):449–458. https://doi.org/10.1109/TCC.2014.2350475
18. Chen J, Wang Y (2019) A hybrid method for short-term host utilization prediction in cloud computing. J Electr Comput Eng 2019. https://doi.org/10.1155/2019/2782349
19. Chen Z, Hu J, Min G, Zomaya AY, El-Ghazawi T (2019) Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans Parallel Distrib Syst 31(4):923–934. https://doi.org/10.1109/TPDS.2019.2953745
20. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555. https://doi.org/10.48550/arXiv.1412.3555
21. Cortez E, Bonde A, Muzio A, Russinovich M, Fontoura M, Bianchini R (2017) Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp 153–167. https://doi.org/10.1145/3132747.3132772
22. Demir S, Mincev K, Kok K, Paterakis NG (2020) Introducing technical indicators to electricity price forecasting: A feature engineering study for linear, ensemble, and deep machine learning models. Appl Sci 10(1):255. https://doi.org/10.3390/app10010255
23. Dezhkam A, Manzuri MT, Aghapour A, Karimi A, Rabiee A, Shalmani SM (2022) A bayesian-based classification framework for financial time series trend prediction. J Supercomput 1–38. https://doi.org/10.1007/s11227-022-04834-4
24. Duggan M, Mason K, Duggan J, Howley E, Barrett E (2017) Predicting host cpu utilization in cloud computing using recurrent neural networks. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, pp 67–72. https://doi.org/10.23919/ICITST.2017.8356348
25. Farahnakian F, Liljeberg P, Plosila J (2013a) Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers. In: 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, IEEE, pp 357–364. https://doi.org/10.1109/SEAA.2013.23
26. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J (2013b) Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In: 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, IEEE, pp 256–259. https://doi.org/10.1109/UCC.2013.51
27. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA (2019) Deep learning for time series classification: a review. Data Min Knowl Discov 33(4):917–963. https://doi.org/10.1007/s10618-019-00619-1
28. Fu X, Zhou C (2017) Predicted affinity based virtual machine placement in cloud computing environments. IEEE Trans Cloud Comput 8(1):246–255. https://doi.org/10.1109/TCC.2017.2737624
29. Gai K, Du Z, Qiu M, Zhao H (2015) Efficiency-aware workload optimizations of heterogeneous cloud computing for capacity planning in financial industry. In: 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, IEEE, pp 1–6. https://doi.org/10.1109/CSCloud.2015.73
30. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. arXiv preprint arXiv:1406.2661. https://doi.org/10.48550/arXiv.1406.2661
31. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, England
32. Guo Y, Yao W (2018) Applying gated recurrent units pproaches for workload prediction. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, IEEE, pp 1–6. https://doi.org/10.1109/NOMS.2018.8406290

33. Hassan HA, Maiyza AI, Sheta WM (2020) Integrated resource management pipeline for dynamic resource-effective cloud data center. J Cloud Comput 9(1):1–20. https://doi.org/10.1186/s13677-020-00212-8
34. Hieu NT, Di Francesco M, Ylä-Jääski A (2017) Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. IEEE Trans Serv Comput 13(1):186–199. https://doi.org/10.1109/TSC.2017.2648791
35. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
36. Huang Z, Peng J, Lian H, Guo J, Qiu W (2017) Deep recurrent model for server load and performance prediction in data center. Complexity 2017. https://doi.org/10.1155/2017/8584252
37. Hu Y, Deng B, Peng F (2016) Autoscaling prediction models for cloud resource provisioning. In: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), IEEE, pp 1364–1369. https://doi.org/10.1109/CompComm.2016.7924927
38. Hu Y, Deng B, Peng F, Wang D (2016) Workload prediction for cloud computing elasticity mechanism. In: 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), IEEE, pp 244–249. https://doi.org/10.1109/ICCCBDA.2016.7529565
39. Hyndman RJ, Khandakar Y, et al (2008) Automatic time series forecasting: the forecast package for r. J Stat Softw 27(3):1–22. https://doi.org/10.18637/jss.v027.i03
40. Hyndman RJ, Athanasopoulos G (2018) Forecasting: principles and practice. OTexts, Australia
41. Janardhanan D, Barrett E (2017) Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, pp 55–60. https://doi.org/10.23919/ICITST.2017.8356346
42. Jozefowicz R, Zaremba W, Sutskever I (2015) An empirical exploration of recurrent network architectures. In: International conference on machine learning, PMLR, pp 2342–2350
43. Kara Y, Boyacioglu MA, Baykan ÖK (2011) Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. Expert Syst Appl 38(5):5311–5319. https://doi.org/10.1016/j.eswa.2010.10.027
44. Karim ME, Maswood MMS, Das S, Alharbi AG (2021) Bhyprec: a novel bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine. IEEE Access 9:131476–131495. https://doi.org/10.1109/ACCESS.2021.3113714
45. Khan T, Tian W, Ilager S, Buyya R (2022) Workload forecasting and energy state estimation in cloud data centres: Ml-centric approach. Futur Gener Comput Syst 128:320–332. https://doi.org/10.1016/j.future.2021.10.019
46. Kim IK, Wang W, Qi Y, Humphrey M (2016) Empirical evaluation of workload forecasting techniques for predictive cloud resource scaling. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), IEEE, pp 1–10. https://doi.org/10.1109/CLOUD.2016.0011
47. Kumar J, Goomer R, Singh AK (2018) Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. Procedia Comput Sci 125:676–682. https://doi.org/10.1016/j.procs.2017.12.087
48. Kumar J, Singh AK, Buyya R (2020) Ensemble learning based predictive framework for virtual machine resource request prediction. Neurocomputing 397:20–30. https://doi.org/10.1016/j.neucom.2020.02.014
49. Kumar M, Kishor A, Samariya JK, Zomaya AY (2023) An autonomic workload prediction and resource allocation framework for fog enabled industrial IoT. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2023.3235107
50. Li C, Bai J, Luo Y (2020) Efficient resource scaling based on load fluctuation in edge-cloud computing environment. J Supercomput 76:6994–7025. https://doi.org/10.1007/s11227-019-03134-8
51. Lin W, Yao K, Zeng L, Liu F, Shan C, Hong X (2022) A GAN-based method for time-dependent cloud workload generation. J Parallel Distrib Comput. https://doi.org/10.1016/j.jpdc.2022.05.007
52. Liu J, Tan X, Wang Y (2019) Cssap: software aging prediction for cloud services based on arima-lstm hybrid model. In: 2019 IEEE International Conference on Web Services (ICWS), IEEE, pp 283–290. https://doi.org/10.1109/ICWS.2019.00055
53. Mahsereci M, Balles L, Lassner C, Hennig P (2017) Early stopping without a validation set. arXiv preprint arXiv:1703.09580. https://doi.org/10.48550/arXiv.1703.09580
54. Melhem SB, Agarwal A, Goel N, Zaman M (2017) Markov prediction model for host load detection and VM placement in live migration. IEEE Access 6:7190–7205. https://doi.org/10.1109/ACCESS.2017.2785280
55. Moghaddam SM, O'Sullivan M, Walker C, Piraghaj SF, Unsworth CP (2020) Embedding individualized machine learning prediction models for energy efficient VM consolidation within cloud data centers. Futur Gener Comput Syst 106:221–233. https://doi.org/10.1016/j.future.2020.01.008
56. Mozo A, Ordozgoiti B, Gómez-Canaval S (2018) Forecasting short-term data center network traffic load with convolutional neural networks. PloS ONE 13(2):e0191939. https://doi.org/10.1371/journal.pone.0191939
57. Nashold L, Krishnan R (2020) Using lstm and sarima models to forecast cluster CPU usage. arXiv preprint arXiv:2007.08092. https://doi.org/10.48550/arXiv.2007.08092
58. Niedermaier S, Koetter F, Freymann A, Wagner S (2019) On observability and monitoring of distributed systems–an industry interview study. In: International Conference on Service-Oriented Computing, Springer, pp 36–52. https://doi.org/10.1007/978-3-030-33702-5_3
59. Ouhame S, Hadi Y, Ullah A (2021) An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. Neural Comput & Applic pp 1–13. https://doi.org/10.1007/s00521-021-05770-9
60. Park K, Pai VS (2006) Comon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Oper Syst Rev 40(1):65–74. https://doi.org/10.1145/1113361.1113374
61. Patel YS, Jaiswal R, Pandey S, Misra R (2020) k stacked bidirectional lstm for resource usage prediction in cloud data centers. In: International Conference on Internet of Things and Connected Technologies, Springer, pp 147–157. https://doi.org/10.1007/978-3-030-76736-5_14
62. Patel YS, Bedi J (2023) Mag-d: A multivariate attention network based approach for cloud workload forecasting. Futur Gener Comput Syst. https://doi.org/10.1016/j.future.2023.01.002
63. Patel M, Chaudhary S, Garg S (2016) Machine learning based statistical prediction model for improving performance of live virtual machine migration. J Eng 2016. https://doi.org/10.1155/2016/3061674
64. Peng C, Li Y, Yu Y, Zhou Y, Du S (2018) Multi-step-ahead host load prediction with gru based encoder-decoder in cloud computing. In: 2018 10th International Conference on Knowledge and Smart Technology (KST), IEEE, pp 186–191. https://doi.org/10.1109/KST.2018.8426104
65. Porambage P, Kumar T, Liyanage M, Partala J, Lovén L, Ylianttila M, Seppänen T (2019) Sec-edgeai: Ai for edge security vs security for edge ai. The 1st 6G Wireless Summit (Levi, Finland). https://link.springer.com/article/10.1007/s10586-021-03492-0. https://link.springer.com/article/10.1007/s13369-021-06348-2
66. Prevost JJ, Nagothu K, Kelley B, Jamshidi M (2011) Prediction of cloud data center networks loads using stochastic and neural models. In: 2011 6th International Conference on System of Systems Engineering, IEEE, pp 276–281. https://doi.org/10.1109/SYSOSE.2011.5966610
67. Sahi SK, Dhaka V (2015) Study on predicting for workload of cloud services using artificial neural network. In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, pp 331–335
68. Saxena D, Singh AK (2021) A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center. Neurocomputing 426:248–264. https://doi.org/10.1016/j.neucom.2020.08.076
69. Selvin S, Vinayakumar R, Gopalakrishnan E, Menon VK, Soman K (2017) Stock price prediction using lstm, rnn and cnn-sliding window model. In: 2017 international conference on advances in computing, communications and informatics (icacci), IEEE, pp 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078
70. Shah J, Vaidya D, Shah M (2022) A comprehensive review on multiple hybrid deep learning approaches for stock prediction. Intell Syst Appl 200111. https://doi.org/10.1016/j.iswa.2022.200111
71. Shaw SB, Singh AK (2015) Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center. Comput Electr Eng 47:241–254. https://doi.org/10.1016/j.compeleceng.2015.07.020

72. Shen S, Van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, pp 465–474. https://doi.org/10.1109/CCGrid.2015.60
73. Shuvo MNH, Maswood MMS, Alharbi AG (2020) Lsru: A novel deep learning based hybrid method to predict the workload of virtual machines in cloud data center. In: 2020 IEEE Region 10 Symposium (TENSYMP), IEEE, pp 1604–1607. https://doi.org/10.1109/TENSYMP50017.2020.9230799
74. Shynkevich Y, McGinnity TM, Coleman SA, Belatreche A, Li Y (2017) Forecasting price movements using technical indicators: Investigating the impact of varying input window length. Neurocomputing 264:71–88. https://doi.org/10.1016/j.neucom.2016.11.095
75. Singh R, Gill SS (2023) Edge ai: A survey. Internet of Things and Cyber-Physical Systems. https://doi.org/10.1016/j.iotcps.2023.02.004
76. Song B, Yu Y, Zhou Y, Wang Z, Du S (2018) Host load prediction with long short-term memory in cloud computing. J Supercomput 74(12):6554–6568. https://doi.org/10.1007/s11227-017-2044-4
77. Tahir F, Abdullah M, Bukhari F, Almustafa KM, Iqbal W (2020) Online workload burst detection for efficient predictive autoscaling of applications. IEEE Access 8:73730–73745. https://doi.org/10.1109/ACCESS.2020.2988207
78. Tschumitschew K, Klawonn F (2017) Effects of drift and noise on the optimal sliding window size for data stream regression models. Commun Stat-Theory Methods 46(10):5109–5132. https://doi.org/10.1080/03610926.2015.1096388
79. Ullah F, Bilal M, Yoon SK (2023) Intelligent time-series forecasting framework for non-linear dynamic workload and resource prediction in cloud. Comput Netw 109653. https://doi.org/10.1016/j.comnet.2023.109653
80. Vashistha A, Verma P (2020) A literature review and taxonomy on workload prediction in cloud data center. In: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, pp 415–420. https://doi.org/10.1109/Confluence47617.2020.9057938
81. Vazquez C, Krishnan R, John E (2015) Time series forecasting of cloud data center workloads for dynamic resource provisioning. J Wirel Mob Netw Ubiquit Comput Dependable Appl 6(3):87–110
82. Wang F, Yu Y, Zhang Z, Li J, Zhen Z, Li K (2018) Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting. Appl Sci 8(8):1286. https://doi.org/10.3390/app8081286
83. Wong JM, Ng ST (2010) Forecasting construction tender price index in Hong Kong using vector error correction model. Constr Manag Econ 28(12):1255–1268. https://doi.org/10.1080/01446193.2010.487536
84. Yang Q, Zhou Y, Yu Y, Yuan J, Xing X, Du S (2015) Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing. J Supercomput 71(8):3037–3053. https://doi.org/10.1007/s11227-015-1426-8
85. Yazdanian P, Sharifian S (2021) E2lg: a multiscale ensemble of lstm/gan deep learning architecture for multistep-ahead cloud workload prediction. J Supercomput 1–31. https://doi.org/10.1007/s11227-021-03723-6
86. Yıldırım DC, Toroslu IH, Fiore U (2021) Forecasting directional movement of forex data using LSTM with technical and macroeconomic indicators. Financ Innov 7(1):1–36. https://doi.org/10.1186/s40854-020-00220-2
87. Zhang W, Li B, Zhao D, Gong F, Lu Q (2016) Workload prediction for cloud cluster using a recurrent neural network. In: 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), IEEE, pp 104–109. https://doi.org/10.1109/IIKI.2016.39
88. Zhu Q, Agrawal G (2012) Resource provisioning with budget constraints for adaptive applications in cloud environments. IEEE Trans Serv Comput 5(4):497–511. https://doi.org/10.1145/1851476.1851516
89. Zhu Y, Zhang W, Chen Y (2019) Gao H (2019) A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. EURASIP J Wirel Commun Netw 1:1–18. https://doi.org/10.1186/s13638-019-1605-z

Publisher's Note