Virtual Machine allocation based on its CPU utilization using Deep Learning

Ketulkumar Polara
KFSCIS
Florida International University
Miami, Florida
kpola009@fiu.edu

Priyaben Barot
KFSCIS
Florida International University
Miami, Florida
pbaro008@fiu.edu

Jose Jimenez
KFSCIS
Florida International University
Miami, Florida
jjime197@fiu.edu

***Abstract*—The allocation of virtual machines (VMs) in virtualized environments is a critical task that requires efficient resource utilization to minimize costs and improve performance. This paper proposes a deep learning-based approach to allocate VMs based on their CPU utilization. We use Long Short-Term Memory (LSTM) algorithm to predict future resource utilization and allocate VMs accordingly. Our proposed solution is evaluated on a real dataset collected from a distributed datacenter. Our results show that our approach outperforms traditional fixed rules-based allocation methods in terms of performance and efficiency.**

***Index Terms*—Virtual Machine, Deep Learning, LSTM, Resource Allocation, CPU Utilization, Virtualized Environment.**

I. Introduction

In recent years, cloud computing has seen tremendous growth and has changed the IT industry, because of its capability to provide practically unlimited scalability, no upfront investment and maintenance for IT infrastructure and usage-based payments. Where optimal Virtual Machine (VM) allocation is a key challenge in cloud computing, as it affects the performance, cost, and energy consumption of cloud providers and users. VM allocation is the process of assigning VMs to physical machines (PMs) in a cloud data center, such that the resource demands of VMs are satisfied by the resource capacities of PMs. VM allocation is a complex and dynamic problem, as it involves multiple objectives and constraints, such as minimizing resource wastage, maximizing resource utilization, balancing the load among PMs, reducing the number of active PMs, satisfying the service level agreements (SLAs) of users, and adapting to the changes in VM requests and resource availability. However, allocating VMs based on fixed rules or pre-defined thresholds for resource utilization can lead to inefficiencies or errors in allocation process. For example, some rules may require global or centralized information, which is hard to obtain or maintain in a large-scale distributed system. Other rules may involve intricate trade-offs or dependencies that are difficult to resolve or balance. The rules may conflict with each other or with other policies or objectives of the cloud provider or the customers which ultimately results in increased costs, and poor performance.

To overcome these problems, some alternative approaches have been proposed, such as using machine learning, optimization, or game theory techniques to dynamically and adaptively allocate VMs. In this paper we propose to solve the subsection of the problem, we will use machine learning techniques like LSTM and CNN-LSTM to forecast the VM's CPU utilization based on VMs historical usage data, the allocation strategy based on forecast results is out of scope for this paper.

We used a dataset of 1594 VMs over 49 hosts from Delft University of Technology (TUDelft) to achieve desired results. We also identified the problem as multivariate forecasting problem. Further some series of preprocessing steps were performed on the acquired data to make the proposed model generalize better followed by model building and measure results using Mean square Error and Mean Absolute Percentage Error.

The proposed deep learning-based approach has the potential to lead to better performance, efficiency, and cost savings in virtualized environments by dynamically allocating VMs based on actual resource utilization our research contributes to the growing body of literature on resource allocation in virtualized environments and highlights the potential of deep learning approaches for addressing the challenges associated with VM allocation.

II. Related Work

The surge in demand for cloud computing services has led to a significant upswing in research in this field in recent years. As a result, various machine learning models have been proposed to predict the CPU utilization of virtual machines (VMs), which play a critical role in efficient resource allocation in cloud environments.

Some of the most widely studied models for predicting CPU utilization include Linear Regression [1], k-nearest neighbor [2], recurrent neural networks [3], and Autoregression neural network (AR-NN) [4]. These models have been evaluated using a range of datasets, such as TPC-W, PlanetLab, and FastStorage, to optimize VM allocation and ensure efficient utilization of cloud resources.

Linear Regression is a basic and widely used statistical model that maps input features to output values linearly. It has been applied to predict CPU utilization of VMs by considering

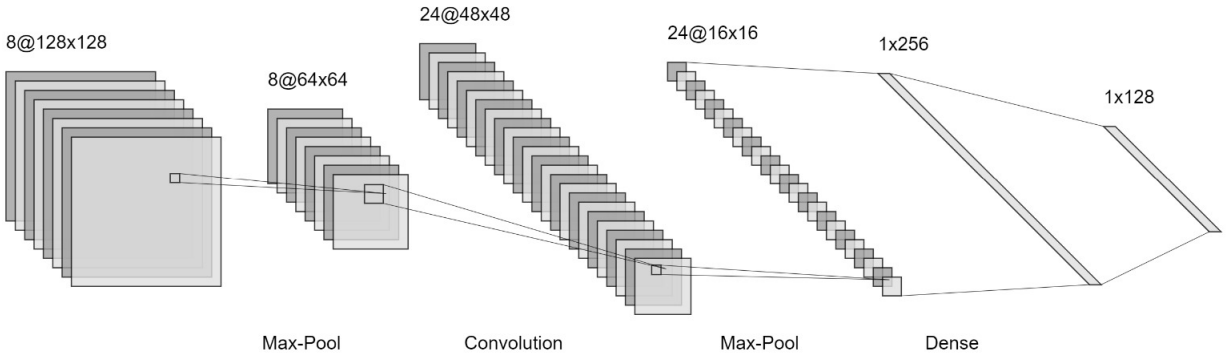

Fig. 1. Structure of convolutional neural network

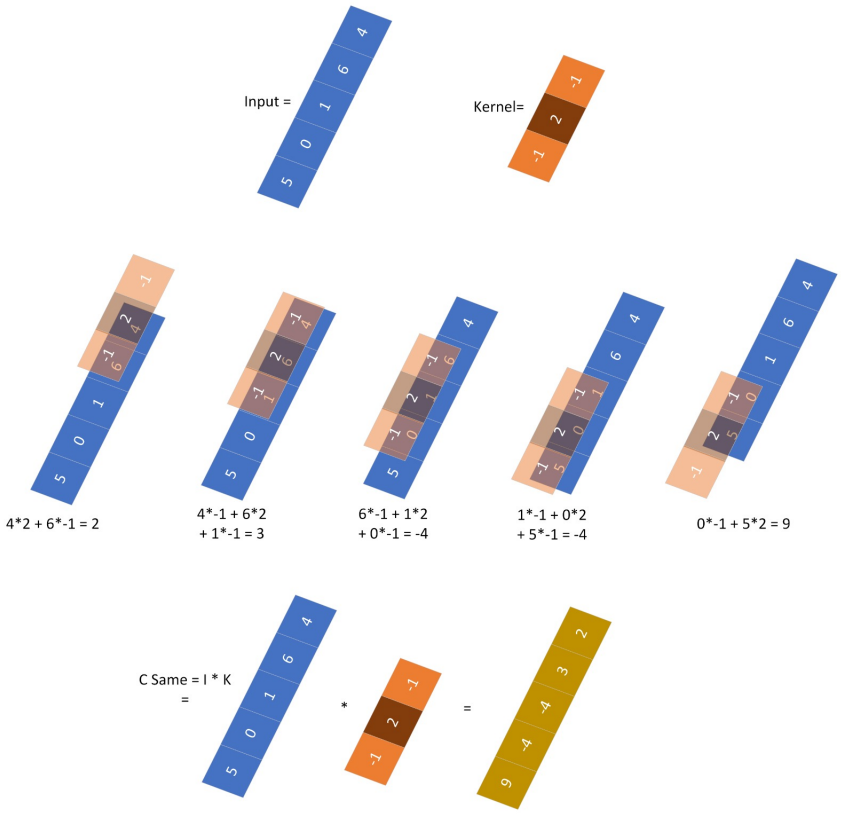

Fig. 2. The operation of the one-dimensional convolution

various features such as disk I/O, network usage, and memory usage.

The k-nearest neighbor (KNN) model is a non-parametric model that works by identifying the k closest data points in the training dataset and using their average values to make predictions. KNN has been applied to predict CPU utilization by considering various features such as disk I/O, network usage, and memory usage.

Recurrent neural networks (RNN) are specialized neural networks that are designed to process sequential data. RNNs have been applied to predict CPU utilization by considering time-series data. They can capture the dynamic dependencies among the time steps and predict the future values of CPU utilization.

Autoregression neural network (AR-NN) is a type of recurrent neural network that uses auto-regressive techniques to model time-series data. It has been applied to predict CPU utilization of VMs by considering past values of CPU utilization as input to the network.

To evaluate and optimize these models for VM allocation,

researchers have utilized datasets such as TPC-W, PlanetLab, and FastStorage. These datasets provide a realistic workload for the models to train on and enable researchers to evaluate the performance of the models under different conditions. Overall, these studies have shown that deep learning models such as RNNs and AR-NNs can provide superior performance for predicting CPU utilization of VMs and can lead to more efficient VM allocation in cloud environments.

III. Methodology

In this paper, we have implemented two different deep learning models: Long Short-Term Memory (LSTM) and Convolution Neural Networks (CNN-LSTM) to forecast CPU utilization of Virtual Machines (VMs), to form optimal VM allocation strategies.

A. Convolution Neural Networks

CNN is a type of feed-forward artificial neural network widely used for image analysis and recognition with powerful feature extraction capability. It extracts features hidden in the input layer using multiple filters or kernels. CNNs use these filters with small receptive fields that slide over the input and produce translation-equivariant response. This allows CNNs to capture local patterns and spatial relationships in data. [5] The other capability of CNNs includes refining and downsampling data dimensionality in time and space which reduces the training parameters and avoids overfitting of the model. The general architecture of CNN model is shown in Figure 1.

A basic convolution neural network architecture usually includes a convolutional layer, pooling layer (ex: MaxPooling and AveragePooling), activation layer and fully connected layer.

The convolutional layer uses a mathematical operation called convolution which applies filters or kernels to an input, to produce an output, such as feature map. The filter slides over the input and performs element-wise multiplication and summation to extract features from the input. Based on the input dimension the dimension of the kernel in convolutional layer is decided by the user. The kernel values are tuned during training using backpropagation algorithm. The mathematical operation of convolution is shown in Figure 2, and it is expressed mathematically as:

$$y_l = w_l * x_l + b_l \tag{1}$$

Pooling layer is a key operation in the convolutional neural network that enhances the feature extraction capability. [6] It aims to reduce the data size of the previous layer and eliminate redundant information. This lowers the number of parameters and computation in the network, which helps to prevent overfitting and improve the model's generalization performance. In this paper, we propose a method that combines max pooling and average pooling to process feature maps. The two pooling operations are illustrated in Figure 3.

Fully connected layer (FCL), which is usually placed at the end of the CNN, acts as a "classifier" in the convolutional neural network. The softmax function is often used in the output layer of a multiclassification problem to ensure that the output value is between 0 and 1, and the sum of all output values is 1. The output value represents the probability of this output, and the one with the highest probability is chosen as the final prediction result.

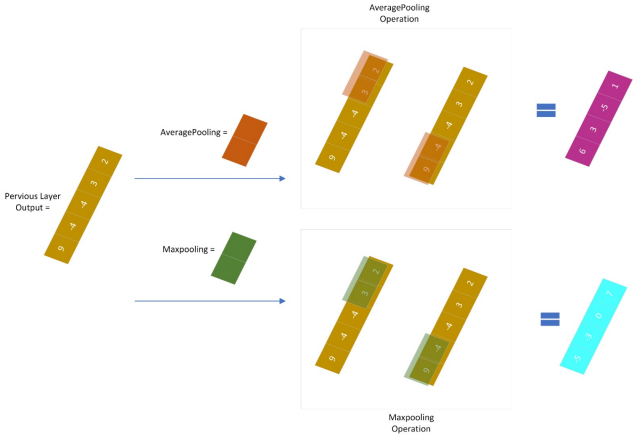

Fig. 3. Schematic diagram of max pooling and average pooling

B. Long Short-Term Memory Network

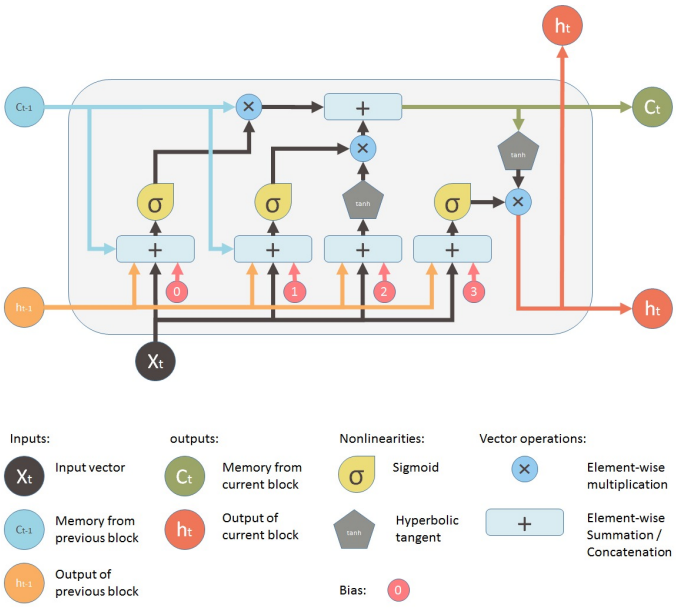

Fig. 4. The internal data operation of the LSTM cell

The recurrent neural network (RNN) is a type of deep neural network that can share information on the time dimension by adding connections (i.e., weights) among neurons in the same layer, making it suitable for dealing with time series problems. [7] Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that has feedback connections and can process not only single data points (such as images), but also entire sequences of data (such as speech or video). LSTM networks are an extension of RNNs mainly introduced

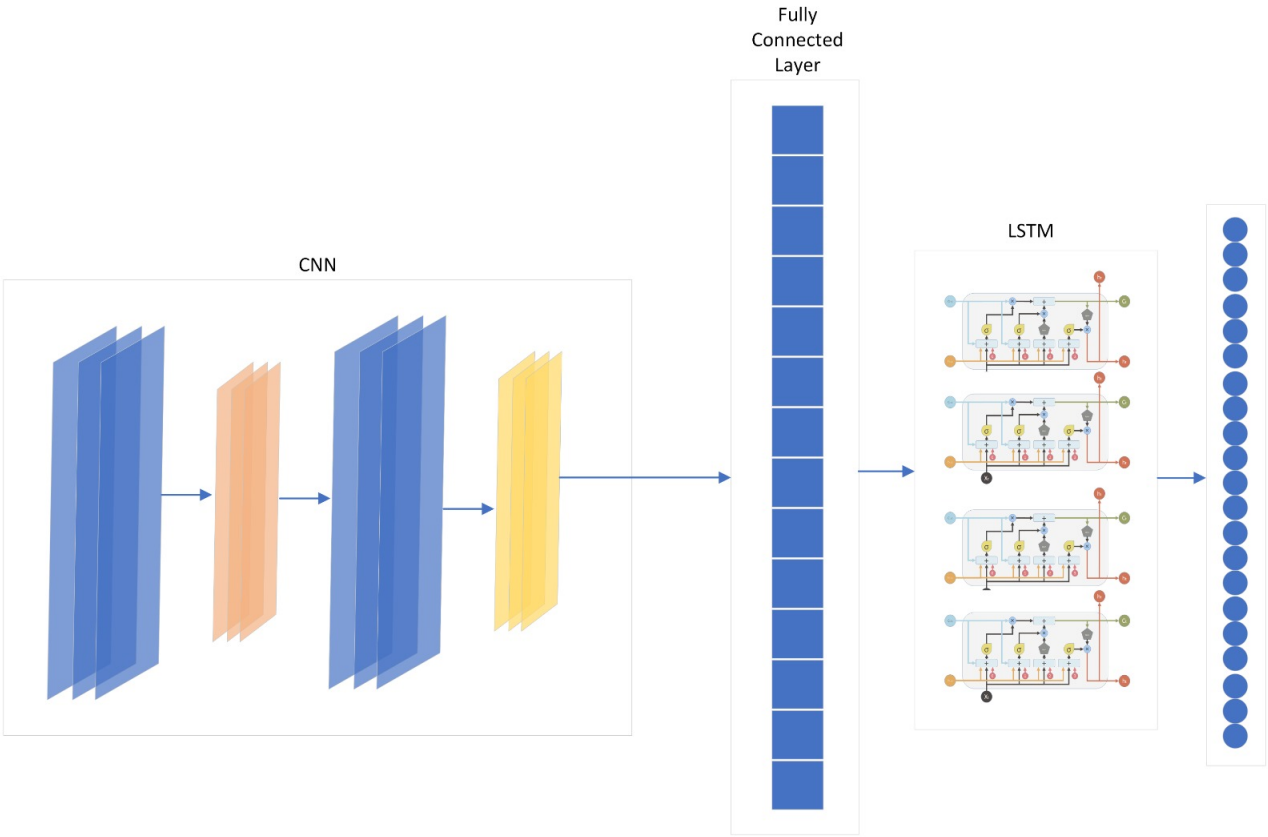

Fig. 5. Structure of the 1DCNN-LSTM model

to handle situations where RNNs fail. LSTM networks have a forget gate that allows the network to forget information that is no longer relevant. This makes LSTM networks more efficient at learning long-term dependencies. The internal data operation of the LSTM cell is shown in Figure 4.

Figure 1 illustrates a recurrent neural network (RNN) cell, which consists of four gates (input, output, forget, and cell gates) and a cell status. The gates use sigmoid activation to determine which information to forget, store, and output. The cell status is updated based on the input and forget gates, and the hidden state is computed using the output gate and the cell status. The input value at time t, denoted by C'_t, is obtained by applying the tanh activation function to the matrix multiplication of the vector $[h_{t-1}, x_t]$ with the weights and biases of the cell status as follows:

$$C'_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2}$$

The forgotten gate f_t, input gate i_t, and output gate o_t at time t are computed using the sigmoid activation function applied to the matrix multiplication of $[h_{t-1}, x_t]$ with the weights and biases of each gate.

$$\begin{aligned} f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \end{aligned} \tag{3}$$

By element-wise multiplication of f_t with the previous cell state C_{t-1}, the information to forget and retain can be determined, and element-wise multiplication of i_t with the current input cell state C'_t determines the information to save and utilize, controlling C'_t. The hidden node state value at time t is obtained by summing these two products. The output at time t is determined by applying the tanh function to the cell status multiplied by the output gate o_t.

IV. Implemented Forecasters

An end-to-end forecasting algorithm based on CNN-LSTM, which forecasts future resource utilization of VMs based on historical usage data, is proposed in this paper.

A. Proposed CNN-LSTM Model

The proposed end-to-end forecasting model based on CNN-LSTM includes stacked convolutional layers, pooling layers, an LSTM layer, and a fully connected layer. The first two pooling layers use max pooling to select the most salient and relevant features, while the last layer uses average pooling to preserve more useful information. The LSTM layer is placed after the fully connected layer to capture the temporal dependency of the features extracted by the convolution operation and fully represent the time series data. The final layer is a dense layer which predicts future values.

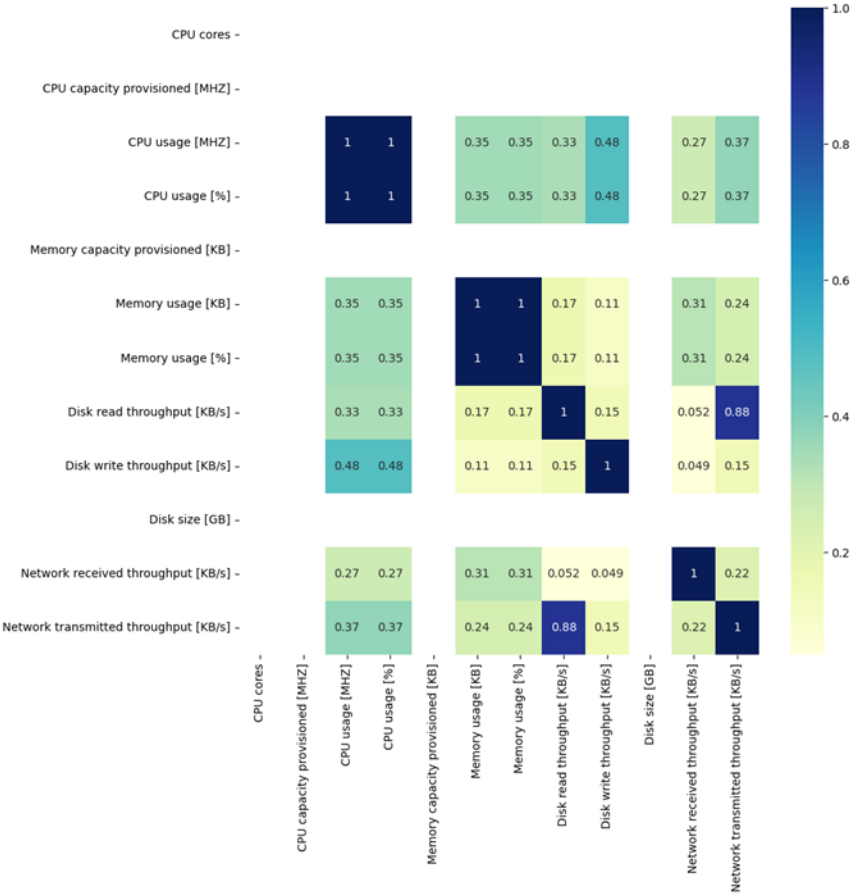

Fig. 6. Correlation Heatmap

B. Forecasting Process

The Forecaster based on CNN-LSTM includes two steps. Firstly, the CNN-LSTM model is built and trained on the training set; Secondly, the model is optimized on validation set and then the model can be used to make predictions future CPU utilization. The Forecasting flowchart of CNN-LSTM is shown in Figure 5. The model building sets:

Step 1: Acquire data from Delft University of Technology (TUDelft) website.

Step 2: Performed data cleaning operations, implement feature selection process using correlation analysis and Normalize data using Standard Normalization.

Step 3: Implement sliding window technique on dataset to prepare data as inputs and labels. In these papers we used the past 48 hours as inputs to predict 1 hour in the future. [8]

Step 4: Divide the dataset into training set, validation set, and test set.

Step 5: Set the initial parameter of the model

Step 6: Train the model on the training set. Tune the model parameter on the validation set using forward propagation and backward propagation.

Step 7: Evaluate the trained model using testing set.

V. Experiment and Results

A. Experiment Conditions

The PyTorch framework by Meta and the python language are used in the experiment of this paper. The experiment program is run on the computer with 2.60GHz Intel(R) Core (TM) i7-10750H CPU and 32 GB memory. We ran two different models LSTM and CNN-LSTM using the forecasting process mentioned in Section 4. We first ran the LSTM model to set the benchmark results on the dataset following we ran CNN-LSTM model to achieve desirable results.

B. Dataset Description and Preprocessing

In this paper, as mentioned above the dataset was acquired from Delft University of Technology (TUDelft) website, the dataset contains usage resource utilization data of 1594 VMs over 49 hosts from a distributed data center of Materna. The dataset came with a separate .csv file for each VM trace. Which included features like Timestamp, CPU Cores, CPU capacity provisioned [MHZ], CPU usage [MHZ], CPU usage [%], Memory capacity provisioned [KB], Memory usage [KB], Memory usage [%], Disk read throughput [KB/s], Disk write throughput [KB/s], Disk size [GB], Network received throughput [KB/s], Network transmitted throughput [KB/s].

We performed correlation analysis on all features for feature selection, through which it was found CPU usage [MHZ] and CPU usage [%], Memory usage [KB] and Memory usage [%], Disk read throughput [KB/s] and Network transmitted throughput [KB/s] are highly correlated to each other hence one of the feature from the feature pair can be dropped and it will help LSTM model to generalized better. The correlation graph can be seen in Figure 6.

C. Results

The performance of the model was measured using mean squared error (MSE) and mean absolute percentage error (MAPE) the fitting of both models can also be seen in Figure 7 and Figure 8.

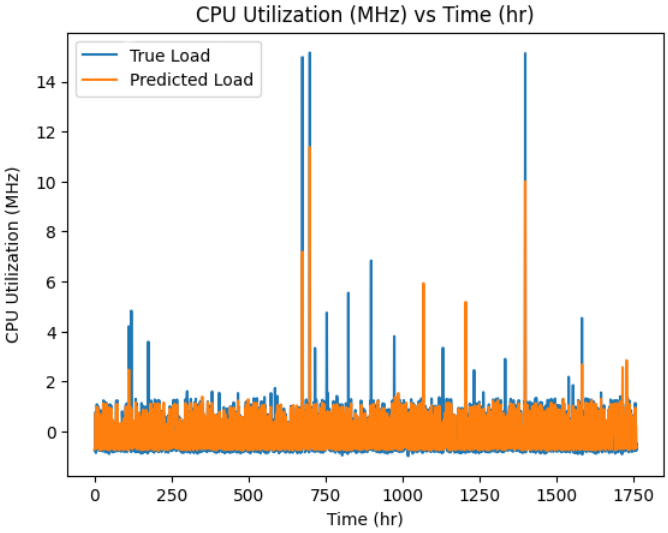

Fig. 7. LSTM

1) Mean Square Error: Mean squared error (MSE) is a measure of the average squared difference between the actual and predicted values in a forecasting analysis.

MSE can be expressed mathematically as follows:

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

where n is the number of observations in the dataset, y_i is the actual value of the target variable for the i^{th} observation, and y^i is the predicted value of the target variable for the i^{th} observation.

2) Mean Absolute percentage Error: Mean absolute percentage error (MAPE) is a metric used to measure the accuracy of a forecasting model or a regression model. MAPE calculates the percentage difference between the predicted and actual values of the target variable, and then takes the average of those percentage differences.

MAPE can be expressed mathematically as follows:

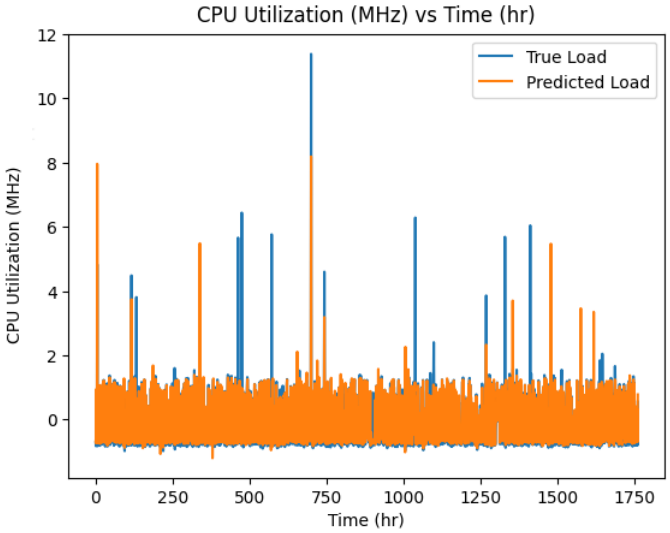

Fig. 8. CNN-LSTM

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}(|(y_i - \hat{y}_i)/y_i)|) \tag{5}$$

where n is the number of observations in the dataset, y_i is the actual value of the target variable for the i^{th} observation, and y^i is the predicted value of the target variable for the i^{th} observation.

Models	MSE	MAPE
LSTM	0.24	0.76
CNN-LSTM	0.22	0.62

ACKNOWLEDGMENT

The authors would like to thank Prof. Dr. Christian Poellabauer for his support and friendly advice and inspiration to this research work.

REFERENCES

[1] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization prediction aware vm consolidation approach for green cloud computing," in *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 2015, pp. 381–388.

[2] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[3] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host cpu utilization in cloud computing using recurrent neural networks," in *2017 12th international conference for internet technology and secured transactions (ICITST)*. IEEE, 2017, pp. 67–72.

[4] Q. Zia Ullah, S. Hassan, and G. M. Khan, "Adaptive resource utilization prediction system for infrastructure as a service cloud," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[5] H. Sun and S. Zhao, "Fault diagnosis for bearing based on 1dcnn and lstm," *Shock and Vibration*, vol. 2021, pp. 1–17, 2021.

[6] Y. LeCun, Y. Bengio, G. Hinton *et al.*, "Deep learning. nature, 521 (7553), 436-444," *Google Scholar Google Scholar Cross Ref Cross Ref*, p. 25, 2015.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] J. Li, Y. Si, T. Xu, and S. Jiang, "Deep convolutional neural network based ecg classification system using information fusion and one-hot encoding techniques," *Mathematical problems in engineering*, vol. 2018, pp. 1–10, 2018.