A COMPARATIVE STUDY OF DEEP LEARNING ARCHITECTURES FOR MULTIVARIATE CLOUD WORKLOAD PREDICTION

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING

By

Derya Gözen

June 2022

A COMPARATIVE STUDY OF DEEP LEARNING ARCHITECTURES FOR MULTIVARIATE CLOUD WORKLOAD PREDICTION

By Derya Gözen

June 2022

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

İbrahim Körpeoğlu(Advisor)

Özgür Ulusoy

Ertan Onur

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

A COMPARATIVE STUDY OF DEEP LEARNING ARCHITECTURES FOR MULTIVARIATE CLOUD WORKLOAD PREDICTION

Derya Gözen
M.S. in Computer Engineering
Advisor: İbrahim Körpeoğlu
June 2022

Cloud computing and use of cloud data centers are in high demand due to their benefits to customers including but not limited to low cost, high availability, reliability, robustness and scalability. Cloud service providers are obliged to fulfill service level agreements that promise high quality of service to their customers. This brings out the need for effective and efficient utilization of data center resources, especially the resources of the compute servers. To achieve proactive and effective resource allocation and scaling policies, accurate prediction of workloads in cloud computing environments plays a critical role. Cloud workload prediction is a challenging task due to high dimensionality, variance and complexity of the workload data. In addition, workload prediction models are expected to work with sufficient amount of past observations to correctly learn workload patterns, at the same time, handle longer forecast horizons accurately. In order to tackle this problem and address the challenges, we investigated and compared five deep learning-based schemes for multivariate time series forecasting to predict the CPU utilization of virtual machines in cloud data centers. The performance of the deep learning schemes is analyzed and compared by using two real-world data sets: Alibaba cluster trace and Bitbrains trace. Our study reveals the relative strengths and weaknesses of the compared schemes for cloud workload prediction. We also observed that, among the compared schemes, Encoder-Decoder LSTM Network with Attention is a more effective solution for workload prediction in cloud computing.

Keywords: Cloud computing, workload prediction, deep learning, multivariate time series, time series forecasting.

ÖZET

ÇOK DEĞİŞKENLİ BULUT İŞ YÜKÜ TAHMİNİ İÇİN DERİN ÖĞRENME MİMARİLERİNİN KARŞILAŞTIRMALI ÇALIŞMASI

Derya Gözen
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: İbrahim Körpeoğlu
Haziran 2022

Bulut veri merkezleri; müşterilere sunulan düşük maliyet, yüksek kullanılabilirlik, güvenilirlik, dayanıklılık ve ölçeklenebilirlik gibi faydalar sebebi ile revaçtadır. Ancak, bulut hizmet sağlayıcıları, hizmet seviyesi anlaşmalarını yerine getirirken, müşterilerine yüksek kalite hizmet vermek ile yükümlüdür. Bu durum, veri merkezi yönetiminde kaynakların etkin ve verimli kullanım ihtiyacını ortaya çıkarmaktadır. Proaktif kaynak paylaştırma ve ölçekleme politikaları elde etme yolunda, bulut ortamlardaki iş yükünün doğru tahmin edilmesi kritik öneme sahiptir. İş yükü tahmini; iş yükü verisindeki yüksek boyutluluk, varyans ve karmaşıklık sebebi ile zorlu bir problemdir. Buna ek olarak, iş yükü tahmin modellerinin, yeterli sayıda geçmiş iş yükü gözlemleri ile çalışması, aynı zamanda geleceğe yönelik uzun tahmin ufku ile başa çıkabilmesi beklenmektedir. Bu problemi çözmek ve zorluklarını irdelemek amacı ile bulut veri merkezlerindeki sanal makinelerin CPU kullanımını tahmin eden, beş farklı derin öğrenme tabanlı çok değişkenli zaman serisi tahmin modeli kurulmuştur. Model performansları iki gerçek dünya veri seti kullanılarak karşılaştırılmış ve analiz edilmiştir: Alibaba küme izleri ve BitBrains izleri. Karşılaştırmalı çalışma ve analiz sonucunda, Dikkat Mekanizmalı Kodlayıcı-Kodçözücü Uzun-Kısa Süreli Bellek Ağının bulut veri merkezlerinde iş yükünün tahmini için etkin ve verimli bir çözüm olduğu görülmüştür.

Anahtar sözcükler: Bulut bilişim, iş yükü tahmini, derin öğrenme, çok değişkenli zaman serisi, zaman serisi tahmini.

Acknowledgement

I would like to express my deepest gratitude to Prof. Dr. İbrahim Körpeoğlu for his acceptance, time, support, invaluable patience, feedback and guidance. I would like to extend my sincere thanks to Prof. Dr. Özgür Ulusoy and Prof. Dr. Ertan Onur for being a part of my defense committee and providing expertise and knowledge.

I am also thankful and grateful to my family, especially my parents, for their moral support. Thanks should also go to my dearest friend Buket Cilalı for always believing in me and lifting my spirit up.

Lastly, words alone cannot express how grateful and thankful I am to have Hakan Maral by my side in this journey as he has shown endless support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the last decade, cloud computing has undergone remarkable changes considering the perspectives of both service providers and customers of these cloud technologies. Compared to the past, the number of cloud providers and amount of services they offer have increased significantly. Application and service providers that were using cloud infrastructures that depend on the data centers of a single provider are now switching to the use of infrastructures from multiple cloud providers [3]. These advances in cloud computing have led to the notion that, one of the most viable computing paradigms, cloud computing, will be recognized as a utility [4].

From an industry-centric point of view, it can be inferred that information technology trends such as cloud computing reshape the way organizations and enterprises operate. They cannot only run their applications stably and faster on the cloud, but also store and process their vast amount of data in external data centers, which significantly decreases the infrastructure costs of the organizations. The lure of cloud computing for the industry is mainly the on-demand services provided in an environment with high availability, reliability and scalability [5].

Consequently, the number of organizations adopting cloud computing increases each day due to the various advantages of cloud computing, including but not limited to low cost, stability, robustness and scalability.

Cloud service providers serve their customers according to service level agreements (SLAs) through huge data centers that act as the base of cloud computing environments. With the application of virtualization concept, data centers play a crucial role in managing the cloud environment more effectively and efficiently. Virtualization enables the creation of multiple virtual machines (VMs), each of which has its own operating system and applications, on a single physical machine [6]. VMs make it possible to share and distribute resources among various users, which is a critical task in cloud computing. Therefore, maximizing the utility of data centers to increase profits on the cloud service providers' side, while providing a high quality of service to customers appears to be a challenge in data center management.

1.2 Problem Definition

Considering the importance of utilizing data center resources very well, an effective resource allocation and scaling policy is required while managing data centers. Although there are various factors that affect resource scaling such as the number of users and the current status of the system [7], provisioning of resources such as central processing unit (CPU), storage (disk space), networking (bandwidth) and memory (RAM) has become a significant aspect in cloud computing. In order to implement a proactive resource scaling policy and improve resource utilization, it is necessary to scale data center resources prior to the arrival of the workloads. Through the accurate prediction of future workloads, resource allocation and scaling can lead to effective and efficient provisioning of resources. Also, power consumption, the number of failed job requests and SLA violations can be reduced in cloud data centers.

Workload prediction in cloud computing is a challenging task due to the following reasons:

- **Historical Workload Data (Past Observations):** In order to obtain an accurate workload prediction model, the model should be able to extract patterns and workload behaviors from historical data. Analysis of the correlation between these patterns is a key step to accurately predict future behaviors. Thus, the amount of historical data required to fit the model needs to be sufficient.

- **High Dimensionality:** Historical workload data for prediction is usually high dimensional. The dimension of data set is proportional to the number of active machines in the data center. For example, if there are m machines in the data center, the workload data set would be m-dimensional and used as input for the workload prediction model [8]. Also, compared to data with low dimensionality, high dimensional data is more likely to be sparse and have noise, which may lead to higher computational cost and error rates in workload prediction.

- **Proactivity:** On-demand provisioning of resources, especially the migration of VMs, can take considerable amount of time [9]. Hence, the workload prediction model is expected to be proactive and provide enough time to manage resources before the arrival of workloads.

- **High Variance:** Workload patterns and behaviors may vary due to the changes in applications hosted, intensity of the workloads and time scales. Some workload patterns are periodic and show high autocorrelation, while other workloads exhibit random fluctuations through time [8]. A workload pattern with high variance makes it difficult for prediction model to learn dynamic behaviors and predict future workloads accurately.

- **Complexity:** Workload prediction model needs to be complex enough to handle high dimensionality and variance of workload data and produce accurate predictions. At the same time, the model needs to operate under the time and cost constraints.

- **Forecast Horizon:** Forecast horizon (prediction length) refers to the length of time for which workload predictions are produced. If it is too long, the model may fail to capture dynamic behaviors of workloads. On the other hand, if it is too short, the model is likely to learn the unimportant information and increase computational cost. Therefore, selecting a proper forecast horizon is essential for the model to learn the most representative workload patterns and generate accurate predictions.

This thesis tackles the problem of workload prediction in cloud computing and introduces a comparative study to address the challenges. More specifically, 5 deep learning-based models are constructed and proposed for workload prediction in cloud computing: Long-Short Term Memory Network, Convolutional Neural Network, Encoder-Decoder Long-Short Term Memory Network, Encoder-Decoder Long-Short Term Memory Network with Attention and Composite Long-Short Term Memory Autoencoder Network. We conduct a comparative analysis based on network architectures and model performance for multivariate time series forecasting.

We performed extensive experiments to determine the strengths and weaknesses of each model under various levels of past observation and prediction length. Two real-world data sets are used for the experiments, which are Alibaba cluster trace data set and Bitbrains trace data set. As a result of our extensive analysis, Encoder-Decoder Long-Short Term Memory Network with Attention is proposed as an accurate and efficient solution to this problem.

The rest of the thesis is organized as follows: Chapter 2 summarizes the related work in the literature for workload prediction in cloud computing. Chapter 3 makes preliminary remarks and introduces the proposed models for multivariate workload prediction. The information about the experiments and the data sets is given in Chapter 4. Next, the results of these experiments are presented and discussed with a comparative analysis in Chapter 5. Finally, Chapter 6 provides the conclusions.

Chapter 2

Related Work

With the advances in modern data centers, provisioning of resources has become a critical task to maximize utility, decrease costs and improve quality of service. One way of resource provisioning is effective and accurate workload prediction in cloud computing. Consequently, workload prediction has drawn researchers' attention and various techniques have been proposed in the literature. These techniques mainly take the historical data of workloads and generate forecasts for the future time steps to effectively provision resources.

Historical data regarding the workloads in cloud computing usually consist of several attributes representing the system status such as CPU and memory usage, disk throughput, in-coming network traffic and out-going network traffic. One or more performance metrics can be used for workload prediction. For instance, there are some studies that consider CPU along with other performance measures such as query per second [10] and RAM [11, 12, 13]. However, a considerable portion of the proposed methods focus only on CPU [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 8]. The reason for commonly using CPU usage for workload prediction problem is that low CPU utility is a vital concern in the clusters of cloud service providers [24, 25].

The studies that tackle the problem of workload prediction in cloud data centers can be roughly grouped into three categories: time series models, machine learning approaches, and deep learning approaches.

2.1 Time Series Models

Hu et al. [26] solve cloud workload prediction problem in order to achieve elasticity in cloud computing. They use some traditional time series analysis methods for workload prediction as well as a Kalman filter model. Their time series approach includes some mathematical models, which are Moving Average (MA), Auto Regression (AR), Auto Regression Integrated Moving Average (ARIMA), difference model and median model. Zhang et al. [12] propose a solution to minimize the energy cost subject to performance constraints and predict the usage of resources using ARIMA. Additionally, the proposed method in [27] uses Auto Regression Moving Average (ARMA) for workload prediction. In an attempt to improve performance, Cao et al. [20] include several prediction models to create an ensemble. In particular, their base models are AR, exponential smoothing, weighted nearest neighbors, and most similar pattern models. Calherios et al. [28] also use ARIMA for workload prediction and assess their results with respect to the improvements and efficiency on resource utilization. Besides, in [29], the authors create an ensemble model to take the advantages of various models including linear models and a machine learning-based neural network. The linear models in this ensemble are MA, weighted average and multi-linear regression.

Time series models appear to be effective for short-term dependencies. However, if time series data has long-term dependency between observations, which is often the case for cloud workloads, these models lack the ability to predict workload for longer time steps into the future accurately. Also, it is difficult for these models to analyze highly-variable data.

2.2 Machine Learning Approaches

Some studies propose clustering-based approaches for workload prediction in cloud computing, where jobs with similar characteristics are grouped in the same cluster. Reference [16] introduces a job-pool based concept and use a neural network architecture to cluster jobs based on their workloads. Then, new jobs are placed into the corresponding clusters to predict the future workload. The authors also discuss the use of ARIMA for prediction and conclude that none of the regression-based methods are able to capture random behaviors in workload patterns (spikes). Another study with a similar approach applies clustering algorithm to group tasks, and predict the workload based on the information each cluster holds [15]. Moreover, Khan et al. [17] develop a co-clustering method to cluster VMs that show correlated behaviors. They also use Hidden Markov Modeling to expose the temporal correlations in clusters and to predict workloads.

Machine learning-based approaches for both workload prediction and classification are also available in the literature. Their main motivation is that there are various machine learning models capable of performing predictions for both regression and classification tasks. A study by Baig et al. [30] proposes an adaptive method to select the best model to accurately predict data center resource utilization. The proposed method consists of a classifier that uses statistical properties of the past resource utilization data.

Additionally, Kaur et al. [18] introduce a model that integrates feature selection and prediction tasks to increase performance. The model is called Regressive Ensemble Approach for Prediction (REAP). It uses genetic algorithm, which is a type of metaheuristic algorithm, for feature selection. For workload prediction, REAP uses an ensemble model generated from a set of 8 machine learning models consisting of Bayesian ridge regression, bidirectional recurrent neural networks, support vector machine, decision tree, extreme learning machine, linear model, neural network and random forest [18]. Another study that uses evolutionary algorithms along with neural networks is by Kumar and Singh [7]. The authors use self adaptive differential evolution algorithm, which is a simple yet successful

technique, to train the neural network and obtain the optimal crossover rate and mutation strategy.

A study by Mason et al. [19] implements various types of evolutionary neural networks to predict workload, which are particle swarm optimization, differential evolution and covariance matrix adaptation evolutionary strategy. They compare the results to the following baseline models: random walk forecasting, MA, linear regression and backpropagation. Moreover, Tang et al. [31] propose a system that is a combination of wavelet neural network architecture and linear regression for workload prediction. With predictions, they deploy a system to schedule jobs efficiently.

Machine learning-based methods for workload prediction seem to revolve around traditional neural network architectures, time series clustering approaches, regression techniques and metaheuristic algorithms. With the rise of deep learning in the last decade, architectures such as Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), Gated Recurrent Units (GRU) seem to be very promising for time-dependent data analysis.

2.3 Deep Learning Approaches

Zhang et al. [11] propose RNN for cloud workload prediction. Although they conclude that the RNN architecture is capable of solving prediction problems for short-term time sequences, it is unable to adapt to long-term prediction. Duggan et al. [14] also use RNN-based method and compare its performance against some traditional methods such as random walk forecasting and MA.

LSTM models are widely used for prediction tasks due to their ability to capture long-term dependencies in a sequential data set. In [22], the authors apply LSTM for adaptive multi-step-ahead workload prediction. Gupta and Dinesh [21] propose LSTM and bidirectional LSTM models for multivariate data analysis and compare their workload predictions with various baseline models including

ARIMA and univariate bidirectional LSTM. Their conclusion is that proposed multivariate LSTM models outperform univariate models and generate more accurate predictions of workloads by capturing long-term dependencies. They also state that bidirectional model is superior to unidirectional model in prediction accuracy. Furthermore, in [10], RNN-LSTM models are proposed for both workload and performance prediction.

GRU is a structure widely used due to its smaller set of parameters (compared to LSTM), and ease of convergence. Guo and Yao [13] present a workload prediction model based on GRUs, which is able to learn temporal patterns and long-term dependencies. Compared to exponential smoothing, ARIMA and LSTM-RNN models, GRU based model shows the lowest error rate. It is also found to be more time-efficient than LSTM-RNN model.

Although deep learning-based methods usually include RNN, LSTM and GRU architectures, Qiu et al. [32] introduce a deep model consisting of a deep belief network to extract high level features, and a regression layer to predict future workloads. Their model outperforms simple artificial neural network, multilayer neural network and ARIMA models.

RNN-based workload prediction models may suffer from time-cost of training due to the high complexity of the architectures. Thus, some methods have been proposed for dimensionality reduction of workload data. These methods are used to obtain the representation of workloads with some of the most important features. Autoencoders appear to introduce the non-linearity of neural networks to reduce dimensionality [8]. In order to handle high dimensionality of workloads in cloud computing, some studies introduce autoencoder-based solutions for workload prediction [8, 23]. Yang et al. [23] present an echo state network with autoencoder and evaluate model performance by implementing several others such as AR and artificial neural network models. In addition, a top-sparse autoencoder is proposed by Chen et al. [8] specifically to address the high dimensional and highly variable workloads in cloud computing. Another model proposed in [33] uses LSTM encoder-decoder architecture with attention mechanism to improve model's ability to predict workloads.

Deep learning-based models are proven to generate more accurate predictions for cloud computing than time series models such as ARIMA, and traditional machine learning approaches such as simple neural network architectures.

Chapter 3

Methods

3.1 Preliminaries

This section briefly introduces the concepts of Long-Short Term Memory, 1D convolution operation and LSTM autoencoders, which are all relevant to the structure of the deep learning-based models.

3.1.1 Long-Short Term Memory

In order to solve problems related to sequential data, LSTMs have been widely adopted due to their ability to capture long time dependencies. In the literature, LSTM-based models have achieved state-of-the-art results for various problems such as speech recognition, natural language processing, audio and video analysis.

Another significant problem that deals with sequential data is time series forecasting. LSTMs are capable of effectively remembering, forgetting or neglecting the sequential data points on probabilistic terms. The commonly used LSTM architecture consists of a cell that can preserve its state, and three gates (*input*, *forget* and *output* gates) that control the information going in and out of the

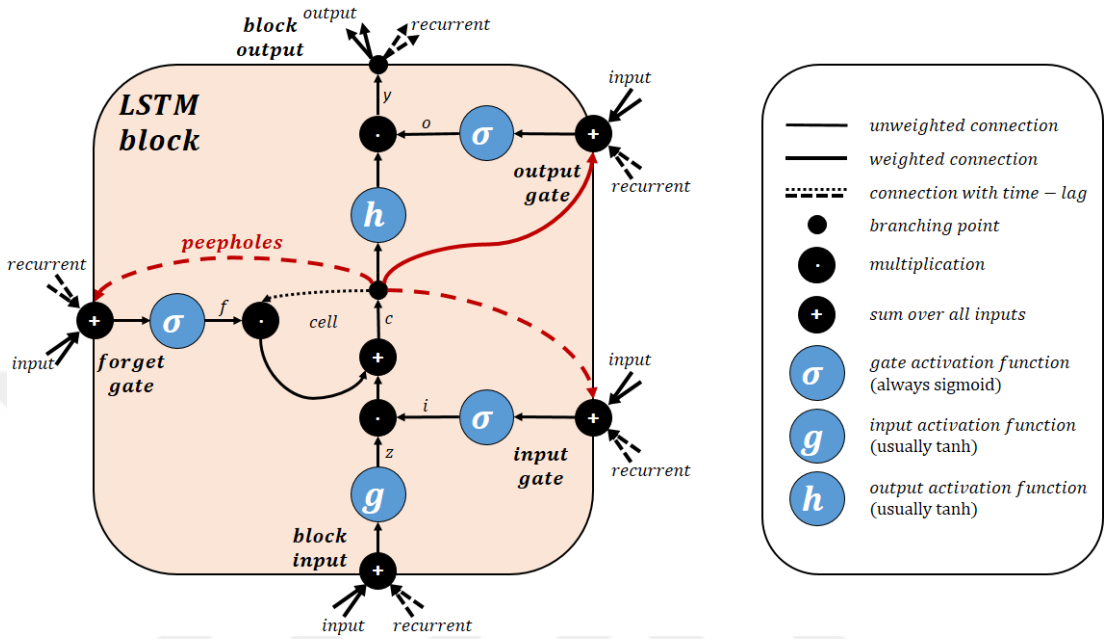


Figure 3.1: A Long-Short Term Memory block (Replica of the figure presented in [1]).

cell [1]. A visual representation of the LSTM cell is in Figure 3.1. Forget gate provokes the learning of sequential tasks with its ability to reset LSTM's state. On the other hand, peephole connections help in learning precise timings easily by connecting the cell to the gates, as shown in Figure 3.1.

Given the input x^t at time t, input weights (W_z, W_i, W_f, $W_o \in \mathbb{R}^{NxM}$), recurrent weights (R_z, R_i, R_f, $R_o \in \mathbb{R}^{NxN}$), peephole weights (p_i, p_f, $p_o \in \mathbb{R}^N$), and bias weights (b_z, b_i, b_f, $b_o \in \mathbb{R}^N$), the block output is calculated as following:

$$
\begin{aligned}
\bar{z}^t &= W_z x^t + R_z y^{t-1} + b_z \\
z^t &= g(\bar{z}^t) \\
\bar{i}^t &= W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i \\
i^t &= \sigma(\bar{i}^t) \\
\bar{f}^t &= W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f \\
f^t &= \sigma(\bar{f}^t) \\
c^t &= z^t \odot i^t + c^{t-1} \odot f^t \\
\bar{o}^t &= W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o \\
o^t &= \sigma(\bar{o}^t) \\
y^t &= h(c^t) \odot o^t
\end{aligned}
\tag{3.1}
$$

where M and N are the number of inputs and the number of LSTM blocks, respectively [1]. In Equation 3.1, the hyperbolic tangent function ($g(z) = h(z) = tanh(z)$) is used as the input and output activation functions, whereas the sigmoid function ($\sigma(z) = \frac{1}{(1+e^{-z})}$) is used as the gate activation.

The weights and bias values are updated according to backpropagation through time.

3.1.2 1D Convolution

Over the last decade, Convolutional Neural Networks (CNNs) have become a primary architecture for the problems encountered in computer vision domain. CNNs consist of convolutional and sampling layers that deal with multi-dimensional data such as images and videos. They are proven to effectively learn complex patterns and extract information from data. While traditional artificial neural network models often follow pre-processing steps or use hand-crafted features, CNNs gather feature extraction and classification in one architecture. This is one of the key aspects that increases the attractiveness of CNNs for complex tasks.

Even though CNNs, i.e., two-dimensional (2D) CNNs, are inherently designed to process multi-dimensional data such as images, recent studies suggest an alternative version of CNN, called one-dimensional (1D) CNN. 1D CNNs can be used for one-dimensional signal processing tasks such as speech recognition and electrocardiogram monitoring [34]. Kiranyaz et al. [35] observe that 1D CNNs might have advantages over 2D CNNs for 1D signal processing tasks. Their observations indicate that compared to 2D CNNs, 1D CNNs have considerably lower computational complexity, use relatively fewer layers in their architecture, are less likely to require special hardware set-up, and more suitable for applications that require real-time processing. In addition to 1D signal processing, some studies propose models based on 1D CNN architectures for time series prediction [36, 37, 38].

Let x be the input of length n that enters the convolutional layer, h be the kernel of length k, and s be the stride of the filter. Then, 1D convolution operation on x with kernel h and stride s is performed as following [39]:

$$y(n) = \begin{cases} \sum_{i=0}^{k} x(n+i)h(i), & \text{if } n = 0. \\ \sum_{i=0}^{k} x(n+i+(s-1))h(i), & \text{otherwise.} \end{cases} \tag{3.2}$$

It should be noted that the output length after convolution operation defined in Equation 3.2 is not the same as the input length. If necessary, input and output lengths can be equalized by using padding technique.

Briefly, 1D CNNs consist of convolutional and pooling layers that process 1D sequence. These layers are capable of interpreting the input and extracting features. Then, extracted features are mapped to a one-dimensional vector by flattening operation. Flattening layer is followed by a fully connected layer for interpretation of the extracted features.

1D CNN architectures can be designed for both univariate and multivariate time series forecasting. When used for time series prediction, these models simply approach time series data as a sequence where 1D convolution operation is performed through. In other words, they do not acknowledge that the data have

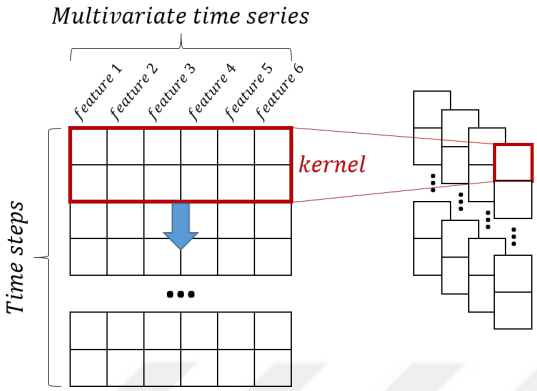


Figure 3.2: Demonstration of 1D convolution operation on time series data. Input and kernel dimensions are all arbitrary.

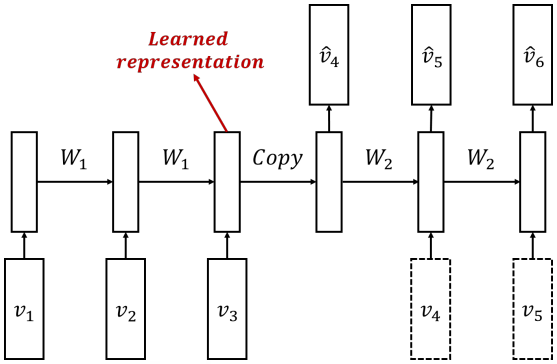
time steps. Figure 3.2 demonstrates how 1D convolution operation operates on a multivariate time series.

3.1.3 LSTM Autoencoders

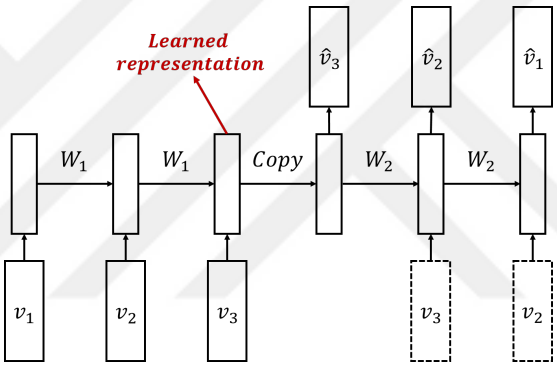
When the problem deals with both the input and output sequences as in translation of a text from a language to another or multi-step time series forecasting, it is referred to as sequence-to-sequence prediction problem. Encoder-Decoder LSTM models are developed in order to unwind such problems by enabling the prediction of variable-length output sequences. In the literature, there are some methods based on encoder-decoder architectures to solve problems related to multivariate and multi-step time series forecasting [40, 41].

Encoder-Decoder LSTM models, as the name implies, are composed of encoder and decoder sub-models. The input sequence is fed into the encoder model for interpretation and output of this model is a vector that represents encoded features. Then, the decoder model takes the output of the encoder model as an input. According to the desired number of output time steps, the decoder model outputs a value for each time step.

Figure 3.3a presents a general architecture of an Encoder-Decoder LSTM model with a single layer.



(a) Encoder-Decoder LSTM model for future prediction.



(b) LSTM Autoencoder model.

Figure 3.3: General architectures of (a) Encoder-Decoder LSTM model and (b) LSTM Autoencoder model [2]. Both models consist of a single layer.

LSTM Autoencoder models are intended to be used for unsupervised learning problems. They comprise two networks: the encoder LSTM and decoder LSTM. Similar to the model described above, encoder of an LSTM Autoencoder model takes and interprets input sequence, then, the output of encoder model is passed through the decoder, where an output is generated for target sequence. Autoencoder LSTM models aim to regenerate the input sequence. Therefore, the output sequence is the same as the target sequence. This differentiates the Autoencoder LSTM models from Encoder-Decoder LSTM models. While Encoder-Decoder LSTM models predict a target sequence into the future, Autoencoder LSTM models tries to predict the input sequence (see Figure 3.3 for comparison).

Figure 3.3b shows a general architecture of LSTM Autoencoder model with a single layer.

3.2 Proposed Models

In this section, five deep learning models are presented to tackle the problem of workload prediction in cloud data centers: Long-Short Term Memory Network, Convolutional Neural Network, Encoder-Decoder Long Short-Term Memory Network, Encoder-Decoder Long Short-Term Memory Network with Attention, and Composite Long-Short Term Memory Autoencoder Network. These models are designed for multivariate and multi-step time series forecasting.

Network diagrams of deep learning models presented in sub-sections 3.2.1, 3.2.2, 3.2.3, 3.2.4 and 3.2.5 are generated using Netron Visualizer [42].

3.2.1 Long-Short Term Memory Network

On the basis of deep learning, the motivation is that hierarchical models with deep architectures are believed to be more efficient than a shallow model as they create new representations at higher levels [43]. Consequently, the success of deep architectures in many challenging problems are often associated with the existence of several layers, each of which processes information and conveys it to the next one [44].

Long-Short Term Memory networks (LSTM networks) are widely used in the literature and proven to be effective for problems related to time-series and sequential data. A standard LSTM network consists of a single LSTM layer, which is followed by a dense layer. In order to take advantage of deep architectures, the depth of LSTM networks can also be increased by stacking LSTM hidden layers with multiple memory cells. Considering the fact that an LSTM network handles sequential data, it can be inferred that using several LSTM layers leads hidden states to operate at different time scales [43]. Therefore, stacked LSTM networks appear to be an efficient technique for time series forecasting and sequence prediction.

The proposed LSTM Network consists of two stacked LSTM hidden layers, followed by a feedforward output layer. The first and second LSTM layers have 512 and 256 units, respectively. The number of units in the output layer depends on the number of features and past observations. As activation function, both LSTM layers use rectified linear units (ReLU). ReLU is a function that outputs 0 if the output is equal to or less than 0; or the value of the input if the input is greater than 0 (Equation 3.3). ReLU has become a widely used activation function for deep neural networks due to its advantages such as fast computation and simplicity.

$$f(x) = max(0, x) \tag{3.3}$$

The proposed LSTM network architecture is shown in Figure 3.4.

3.2.2 Convolutional Neural Network

As discussed in Section 3.1.2, due to the success of 2D CNNs in prediction and classification problems, an alternative convolution operation called 1D convolution is proposed for time series forecasting and classification problems. Whereas 2D CNNs handle multi-dimensional data such as images and videos, 1D CNNs are designed to operate on time series data with a fixed window length.

Convolutional Neural Networks, i.e., CNNs, can be used for multivariate multi-step time series forecasting by adopting the use of 1D convolution operation. 1D CNNs traditionally consists of a convolutional hidden layer where 1D convolution operation is performed over sequential data. The number of convolutional hidden layers might be increased if the input sequence is excessively long. Convolutional layers are followed by a pooling layer, where the feature maps are summarized and down-sampled. Then, the output of pooling layer is flattened to be passed into a fully connected layer. After the dense layer interprets the features extracted by the previous convolutional layer(s), another dense layer produces the output of the model.

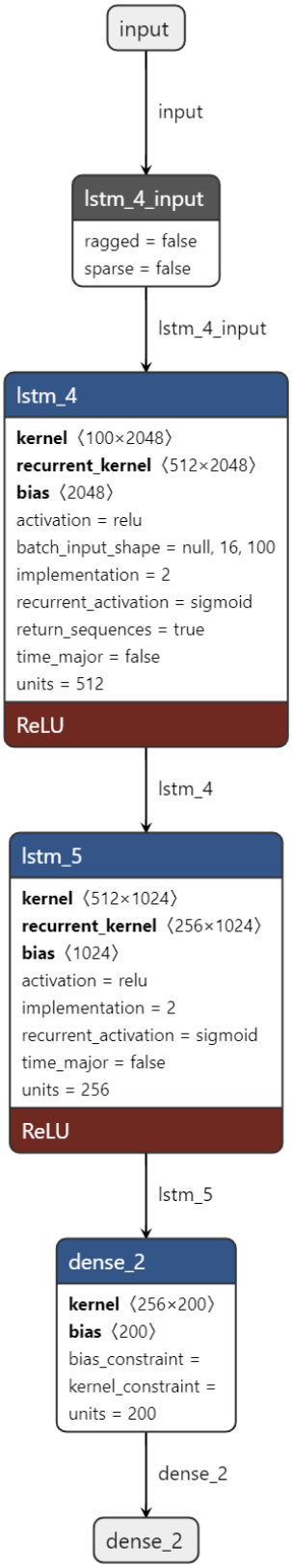


Figure 3.4: LSTM network architecture.

Since the model is not concerned with very long input sequences, the proposed model has a single 1D convolutional layer with 256 filters and a kernel size of 3. The activation function in the 1D convolutional layer is ReLU. Next, there is a max-pooling layer with a pool size of 2. The output of the convolutional part is transformed into a vector representation in the flattening layer. This vector is then passed into the fully connected layer with 128 hidden units and ReLU activation function.

The proposed CNN network architecture is shown in Figure 3.5.

3.2.3 Encoder-Decoder Long-Short Term Memory Network

The recurrent neural network encoder-decoder architectures are specifically designed for sequence-to-sequence problems such as image captioning and machine translation, where state-of-the-art results are achieved. Encoder-decoder architectures are made of two sub-models, called encoder and decoder. The encoder sub-model takes a variable-length input sequence and maps it to a fixed-length vector, whereas the decoder reads this fixed-length vector representation and maps it back to a variable-length output sequence [45].

Encoder-decoder architectures can be constructed with LSTM layers for time series forecasting. The encoder sub-model can contain one or more LSTM layers. The output of the encoder is a fixed-length vector, whose size depends on the number of hidden units (memory cells) in the LSTM layers. The vector that carries the representation of the variable-length input sequence is mapped to the target output sequence in the decoder part.

In the proposed architecture of our Encoder-Decoder LSTM Network, each of the sub-models, the encoder and the decoder, consists of a single LSTM layer, where the number of hidden units is 512. The activation function in LSTM layers is ReLU.

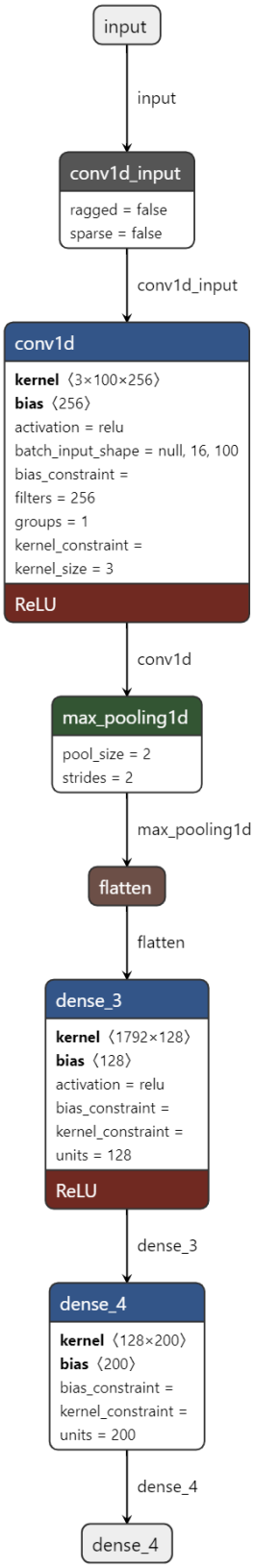


Figure 3.5: CNN network architecture.

The architecture of Encoder-Decoder LSTM Network is presented in Figure 3.6. As can be seen from this figure, the *RepeatVector* acts as the bridge that connects the encoder part of the model to the decoder part by repeating the fixed-length encoded representation of the input sequence. The number of units in the *RepeatVector* layer depends on the prediction length (forecast horizon). Then, in order to predict the target output sequence, a time distributed dense layer is used. The size of inputs and outputs going through each layer is presented elaborately in Figure 3.7.

3.2.4 Encoder-Decoder Long-Short Term Memory Network with Attention

The encoder-decoder architectures are proven to be effective for sequence-to-sequence problems such as machine translation, as discussed in the previous sections. In an encoder-decoder network, the information in input sequence is compressed and represented as a fixed-length vector. This causes a potential issue as the network may face difficulties when dealing with long sequences [46]. In order to overcome this issue and improve the performance of encoder-decoder architectures against long input sequences, attention mechanisms are introduced for recurrent neural networks.

Attention mechanism generates a more beneficial encoding from the input sequence for the construction of the context vector, which is processed by the decoder later on. For example, in sequence-to-sequence problems such as machine translation, attention mechanism helps the model learn the words that require more attention as well as the degree of attention, while predicting each output word in the target sequence. Although attention mechanism is initially proposed by Bahdanau et al. [46] for neural machine translation, Luong et al. [47] seek to increase the effectiveness and simplicity of this approach by introducing some improvements. Therefore, Encoder-Decoder LSTM Network introduced in the previous section is enriched with the attention algorithm proposed by Luong et al. [47].

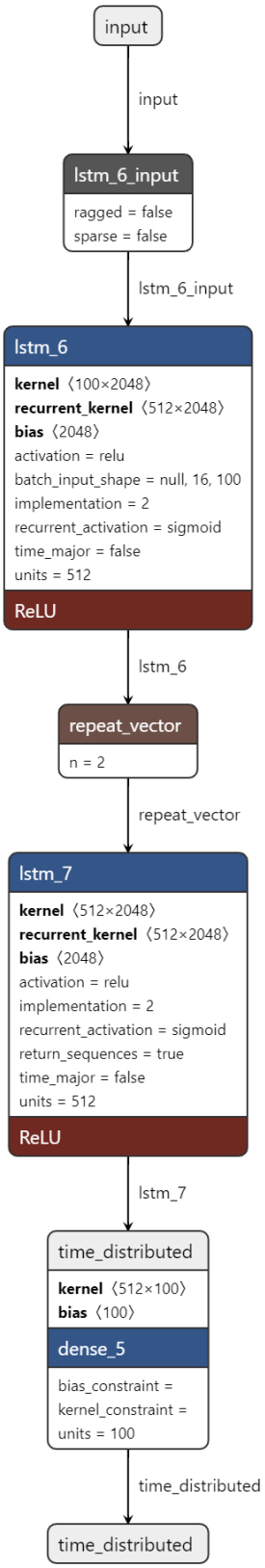


Figure 3.6: Encoder-decoder LSTM network architecture.

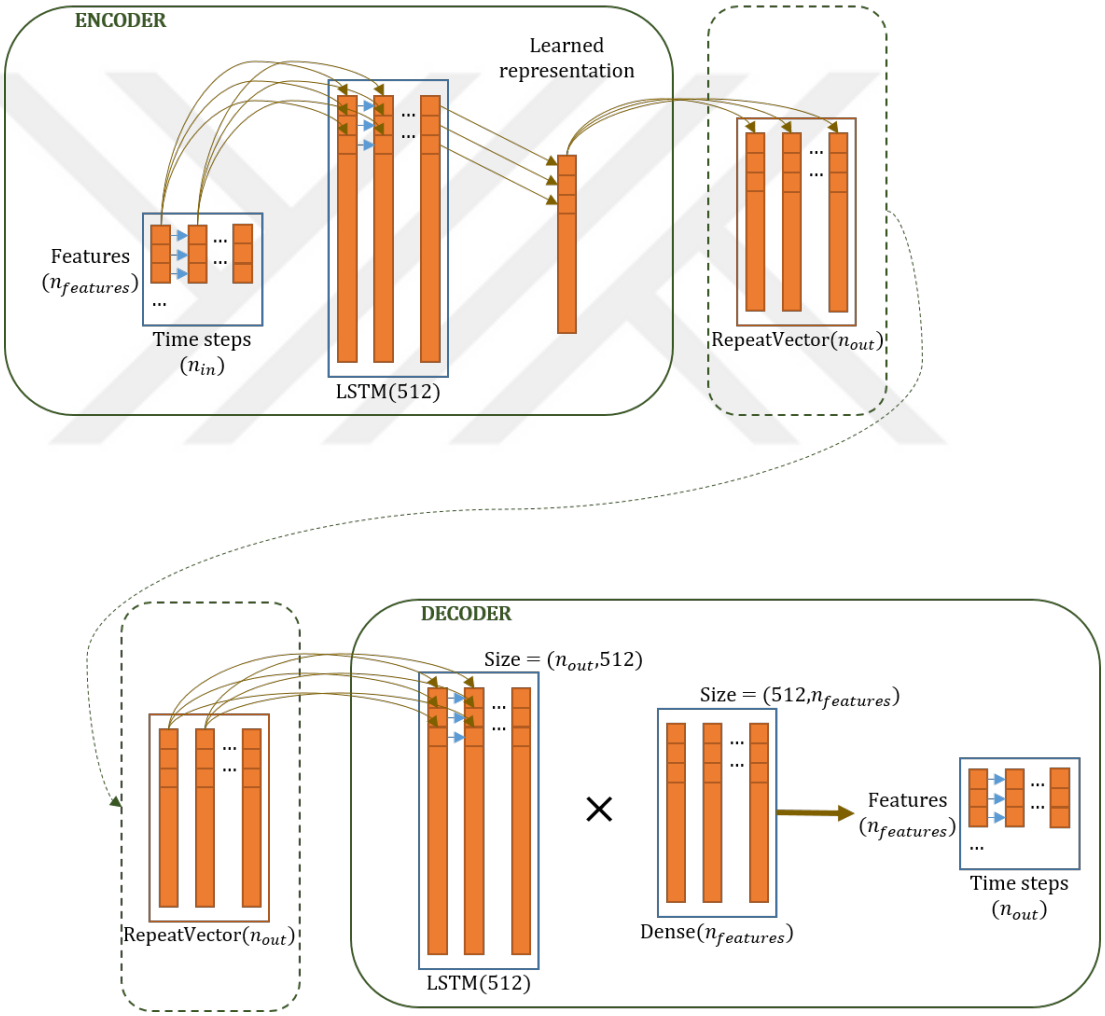


Figure 3.7: A detailed representation of encoder-decoder LSTM network layers.

The attention algorithm first uses the hidden state h_t at time step t (during decoding) to generate a context vector c_t. c_t possesses the information regarding the input sequence that is useful while predicting the current target output y_t. The target hidden state h_t is computed as following:

$$h_t = LSTM_{decoder}(h_{t-1}, y_{t-1}) \tag{3.4}$$

where h_{t-1} and y_{t-1} denote the previous hidden state and the previous output, respectively.

Next, an alignment vector is generated using the current hidden state h_t and source hidden states $\bar{h}_s$ (annotations generated by the encoder). Alignment vector is variable-length vector denoted by a_t and its size equals to the number of time steps in the input sequence.

$$\begin{aligned} a_t(s) &= align(h_t, \bar{h}_s) \\ &= softmax(score(h_t, \bar{h}_s)) \\ &= \frac{exp(score(h_t, \bar{h}_s))}{\sum_{s'} exp(score(h_t, \bar{h}_{s'}))} \end{aligned} \tag{3.5}$$

In Equation 3.5, *score* is defined as a *content-based* function and Luong et al. [47] consider three alternatives for this function. In this study, the *dot* method is used, which is defined as following:

$$score(h_t, \bar{h}_s) = h_t{}^{\intercal} \bar{h}_s \tag{3.6}$$

The following step is to compute the context vector, c_t, as the weighted average of the alignment vector, a_t, over the source hidden states. Given the context vector and hidden state, an attentional hidden state is derived by concatenating the vectors.

$$\tilde{h}_t = tanh(W_c[c_t; h_t]) \tag{3.7}$$

Finally, the decoder part of the network generates the final output by feeding the attentional hidden state to a weighted hidden state.

The architecture of Encoder-Decoder LSTM Network with Attention can be seen in Figure 3.8.

3.2.5 Composite Long-Short Term Memory Autoencoder Network

For sequence-to-sequence problems, LSTM autocoder-based architectures are mostly used to perform two tasks: reconstruction of the input sequence, and future sequence prediction. LSTM autoencoders are generally used for reconstruction, whereas encoder-decoder LSTM models are specifically designed for future prediction. These two tasks can be combined and simultaneously performed within a composite model [2]. Composite models are expected to diminish the drawbacks of both LSTM autoencoders and encoder-decoder architectures. For example, an LSTM autoencoder is likely to memorize the input sequences, which is not desirable for future sequence prediction problems. Another example is that LSTM encoder-decoder architectures tend to give more importance to a few of the last values in the input sequence. Therefore, the encoded feature representation stored in the fixed-length vector may not capture a large portion of the input sequence [2].

Substantially, the encoder part in a composite LSTM autoencoder model is expected to produce feature representations that can be used for both reconstructing input sequences and predicting future sequences. On the other hand, there are two decoders, one of which is responsible for reconstruction, and the other for sequence prediction. A general model diagram for the composite model can be seen in Figure 3.9.

A composite LSTM autoencoder model is proposed for workload prediction in cloud data centers. The encoder part of the network has a single LSTM layer with 512 memory cells. Then, the output of the encoder is passes into the two

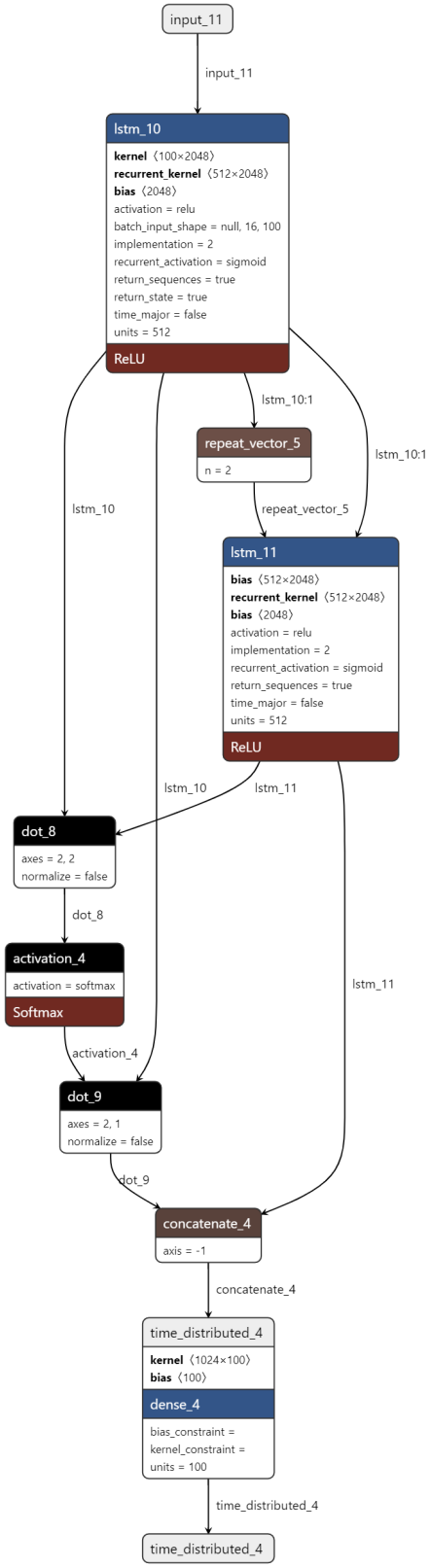


Figure 3.8: Encoder-decoder LSTM with attention network architecture.

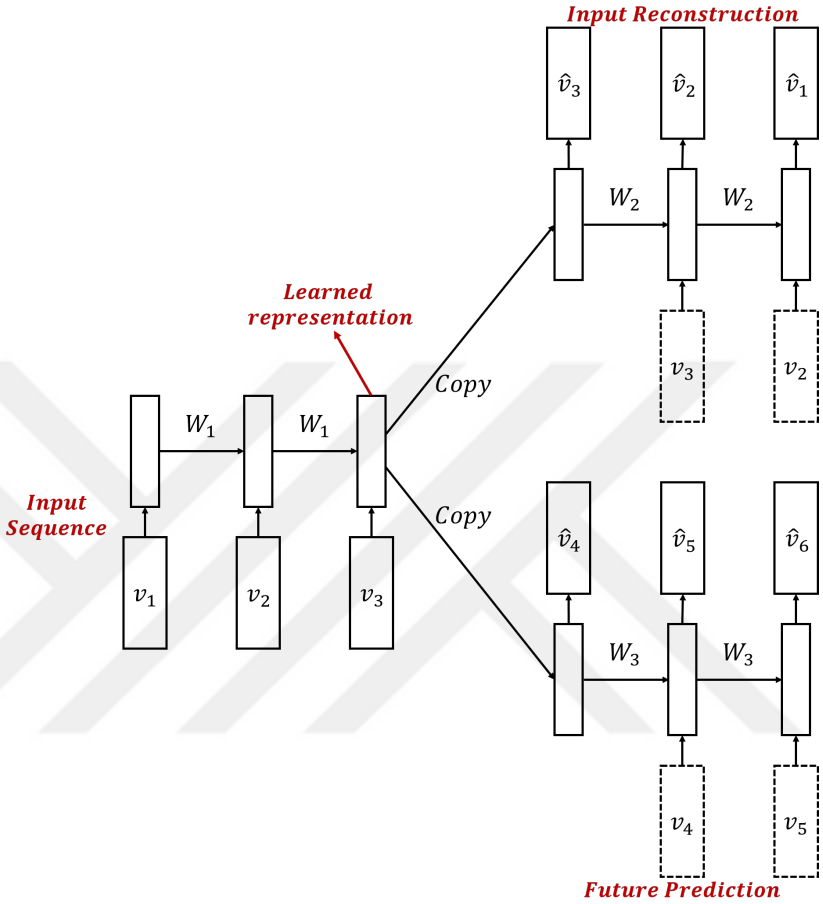


Figure 3.9: General architecture of a composite LSTM autocoder model [2].

decoders with the help of *RepeatVector* layers. Since one decoder is responsible for reconstruction and the other one for prediction, the number of units in these layers is bound up with the number of past observations and the prediction length, respectively. Each of the LSTM layers in the decoders has 512 hidden units, and is followed by a time-distributed dense layer.

The network diagram of the proposed Composite LSTM Autoencoder is presented in Figure 3.10.

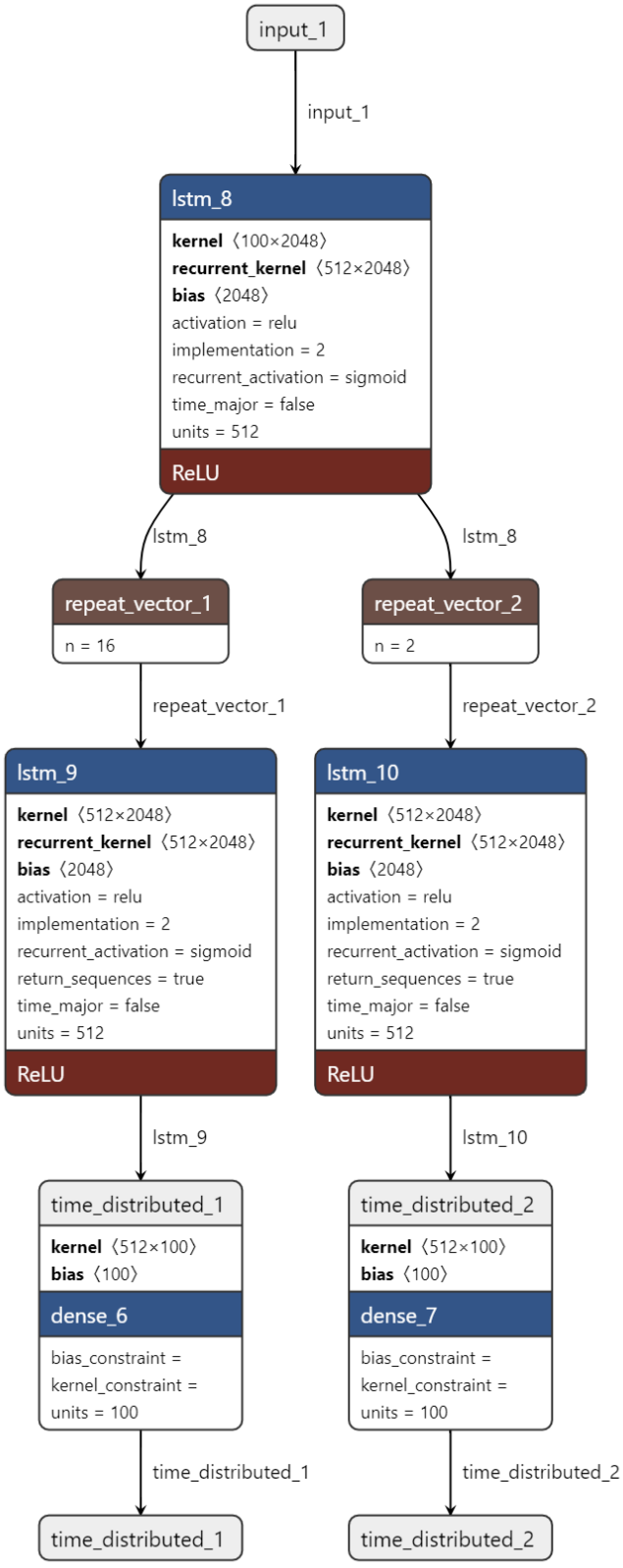


Figure 3.10: Composite LSTM autoencoder network architecture.

Chapter 4

Experiments

4.1 Evaluation Metrics

In order to evaluate the accuracy of the workload prediction, Mean Squared Error (MSE) is reported for each of the proposed models introduced in Section 3.2.

MSE is a widely used evaluation metric for regression problems. Considering the problem in question, that is workload prediction in cloud computing, MSE assesses the performance of workload forecasting models.

MSE is calculated as following:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

where y_i and $\hat{y}_i$ refer to the ground truth value and predicted value of the workload, respectively. The number of time steps predicted by the model is denoted by N.

MSE results in a positive error value as it takes the square of the difference between ground truth and predicted values. Squaring the difference also causes

the models with larger errors to be punished more as larger errors exaggerate the average error value. These properties of MSE make it a proper evaluation metric for regression problems.

4.2 Data Sets

4.2.1 Alibaba Cluster Trace Data Set

Alibaba cluster trace, *cluster-trace-v2018* [48], is a data set released as a part of Alibaba Open Cluster Trace Program and collected from one of Alibaba's production clusters. *cluster-trace-v2018* contains the information about $\sim$ 4000 machines over a period of 8 days, including but not limited to instances in the batch workloads, resource usage of each machine and container.

From Alibaba cluster trace data set, 100 machines are selected randomly, and CPU utilization percentages of these machines are extracted for workload prediction models. Figure 4.1a shows the CPU utilization of two sample machines from Alibaba cluster trace data set. It can be inferred that CPU utilization fluctuates highly and shows random patterns and generally does not show any periodicity or correlation.

4.2.2 Bitbrains Trace Data Set

Bitbrains trace data set is provided by Bitbrains IT Services Inc. and contains the measurements related to the performance metrics of 1750 VMs from Bitbrains data center [49]. From this data set, *fastStorage* trace offers the records of 1250 VMs connected to fast SAN (storage area network) storage devices. The trace data includes information about CPU cores, CPU usage, amount of CPU requested, memory usage, amount of memory requested, disk throughput, and network throughput. The related parameters are sampled every 5 minutes over

a period of one month.

In Bitbrains trace data set, a considerable number of machines have very low CPU utilization, which is $\sim 0\%$. Utilization patterns of such machines tend to be constant and accumulated on the lowest part of the utilization interval. This property of the Bitbrains trace data set makes it difficult to predict workload as machines with such low utilization fail to show time-dependent patterns. Moreover, it can be observed that CPU utilization of a machine notably fluctuates over time, as can be seen from Figure 4.1b. In order to exclude machines that would hinder an accurate prediction model, a two-step filtering operation is performed:

- First, if the fraction of non-zero CPU utilization value of a VM is less than 25%, the VM is excluded, similar to the filtering step in [50].
- Secondly, VMs with average CPU utilization less than 30% are excluded.

The same approach for Alibaba cluster trace data set is adopted for Bitbrains trace, that is random selection of machines for workload prediction. Since two-step filtering operation decreases the number of machines available for prediction models, 50 machines are chosen from Bitbrains trace data set.

4.2.3 Data Preparation

In the scope of this study, workload prediction problem is tackled by considering various deep learning models. Multivariate time series data of clusters first undergoes pre-processing steps to remove noise, and then is re-portrayed as a supervised learning problem for the deep learning models.

For workload prediction, CPU usage is the target indicator of workloads, so let it be defined by the following sequence: $X = (x_1, x_2, x_3, \ldots, x_t)$, where x_t is the value of the CPU usage at time step t.

First of all, in order to prepare data for time series prediction, CPU utilization

percentage data is resampled in a way that the period of sampling is decreased to order of minutes. Next, moving average is applied to the time series as a a smoothing technique. Moving average smoothing is useful in removing the fine-grained noise between time steps and better reveal the structure of the time series.

The next pre-processing step is to normalize CPU usage data to range between $[0, 1]$, according to Equation 4.2.

$$X^{'} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.2}$$

Figure 4.2 shows the raw and pre-processed workload data for two samples of machines from Alibaba cluster trace, whereas Figure 4.3 shows those from Bitbrains trace.

Secondly, after the pre-processing stage, the data is re-framed so as to be in a suitable form for the deep learning models. Since there are multiple sequences of time series data, each of which refers to the CPU utilization of a machine, multiple values are observed at each time step. The multivariate time series data is restructured by creating pairs of input and output samples of sequences. More specifically, previous time steps are used as input (x), whereas next time steps are used as output (y) variables.

4.3 Experimental Setup

The proposed models are implemented for workload prediction in cloud computing and applied on two real-life data sets: Alibaba cluster trace and Bitbrains trace data sets. 100 machines are randomly selected from Alibaba data set for workload prediction and performance indicators are extracted. The selected number of machines is 50 for Bitbrains data set due to unrepresentative workload patterns. Both data sets are challenging as they are high dimensional and show

highly variable workload patterns over time.

CPU utilization is considered to be the fundamental performance indicator of workloads in cloud data centers. The raw data is prepared for deep learning models according to the pre-processing steps explained in Section 4.2.3. Then, data set is split into three portions: training set (60%), validation set (20%) and test set (20%). Training set is used to train the model and calculate the model parameters such as weights and bias terms. While selecting the hyperparameters of the model, validation set assures an impartial evaluation of model fit on training set. On the other hand, the test set is used to objectively evaluate the fit of the final model on the training set.

During the preparation of time series data for deep learning-based models, workload data needs to be re-framed in a way that past and future time steps are paired as input (x) and output (y) variables, respectively. Whether the time series data is univariate or multivariate, there are two important factors to consider while re-framing time series as supervised learning problem:

- **Historical Workload Data (Past Observations):** The number of past observations to be used as input (x).
- **Forecast Horizon (Prediction Length):** The number of steps into the future to be used as output (y).

The number of past observations is an important factor that affects the accuracy of workload prediction model because the model is expected to learn the patterns of historical workloads. Another key factor is the length of prediction, which needs to cover a proper horizon for the model to capture representative patterns. In other words, prediction length should not be too long to prevent the model from learning the behavior of workloads, nor too short to cause the model to learn only the unimportant information about workload patterns. These two factors play a critical role in workload prediction and are among the challenges. In order to better evaluate deep learning-based models, we conducted experiments using various numbers of past observations and prediction length. More

specifically, the set of values is as follows:

$$n_{in} \in \{8, 16, 32, 64\}$$
$$n_{out} \in \{2, 3, 4, 5\}$$

where n_{in} is the number of past observations, and n_{out} is the prediction length.

We use adaptive moment estimation algorithm, Adam, for stochastic optimization. Adam is an optimization algorithm introduced by Kingma and Ba [51] and is based on the gradient descent optimization algorithm. It is built to improve and speed up the optimization process by computing individual adaptive learning rates for the parameters being optimized. This includes calculating the estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients, both of which are decayed exponentially.

Given that the differentiable objective function $f(\theta)$ with parameters θ; step size (learning rate) α; the first and second moments at time step t, m_t and v_t; exponential decay rates for the first and second moment estimates β_1 and β_2, where $\beta_1, \beta_2 \in [0, 1)$, according to the authors, the first step of updating parameters is to calculate the gradients at time step t (for one iteration) as follows [51]:

$$g_t = \nabla_\theta f_t(\theta_{t-1})$$

Next, the estimates of the first moment and second raw moments of the gradients are updated. Initial values of the first and second moments are zero ($m_0 = 0, v_0 = 0$).

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * {g_t}^2$$

Then, bias-corrected first and second moments are calculated. It should be noted that the hyperparameters β_1 and β_2 are decayed over the course of iterations.

$$\hat{m}_t = m_t / (1 - {\beta_1}^t)$$
$$\hat{v}_t = v_t / (1 - {\beta_2}^t)$$

Finally, the parameters are updated according to the following equation:

$$\theta_t = \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

where ϵ is a very small value that is inserted to prevent division by zero.

Regarding the training configuration, the number of epochs is set to 100, the learning rate to 0.0001, and batch size to 256. Since workload prediction is a regression problem, MSE is preferred as the loss function.

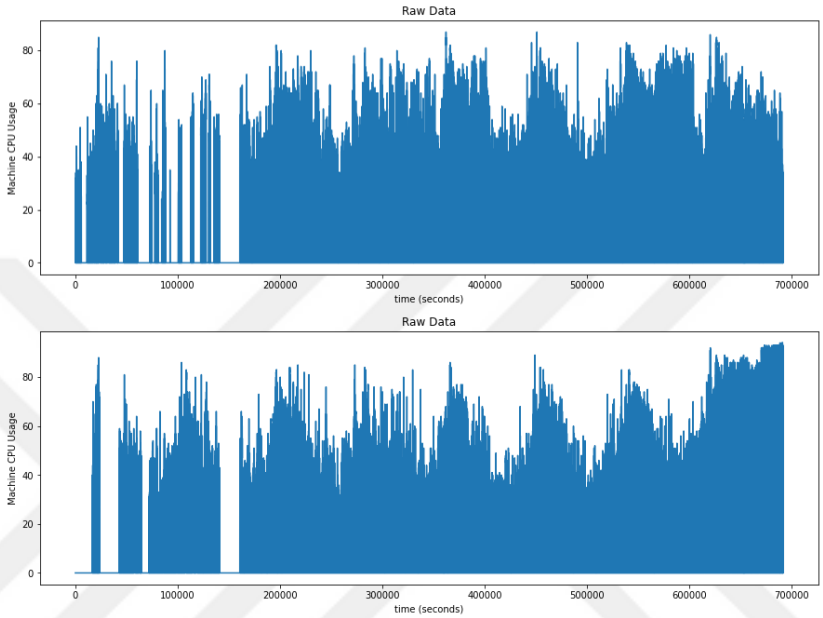
The weights are initialized using Glorot uniform initializer, also known as Xavier uniform initializer. Glorot uniform initializer chooses samples from a uniform distribution bounded by:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{fan_{in} + fan_{out}}}, \frac{\sqrt{6}}{\sqrt{fan_{in} + fan_{out}}}\right] \tag{4.3}$$

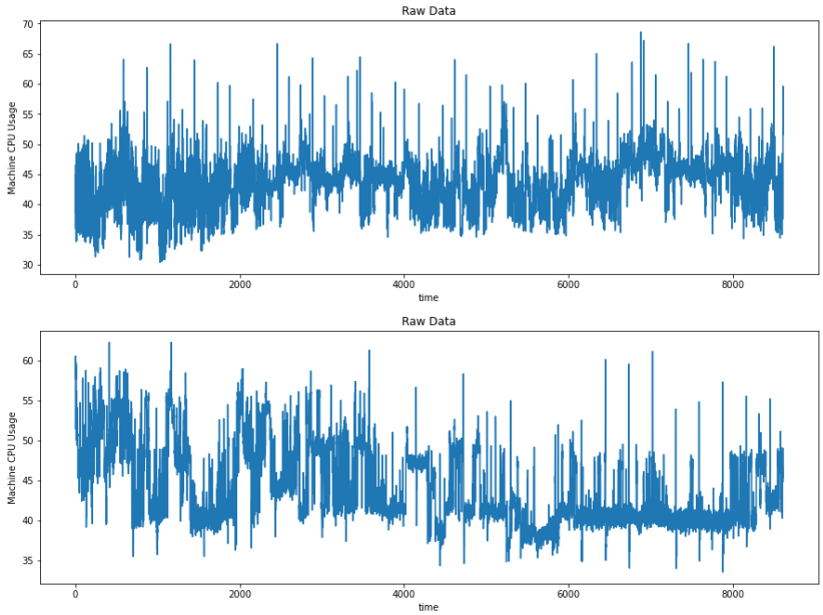
where fan_{in} and fan_{out} are the number of input and output units [52]. The biases are initialized to zero.

Furthermore, while training deep learning-based models, choosing the number of epochs is a critical step since a very large number of epochs may cause overfitting, whereas a very small number of epochs may lead to underfitting. In order to reduce overfitting, an early stopping method is used while training each deep learning model. More specifically, the loss on validation set is monitored during training, which takes 100 epochs. The model that minimizes the loss on validation set is said to be the best model for that particular training session and saved as the final model.

The experiments are executed on a computer with Intel® Core™ i7-6700 CPU @ 3.40GHz, 16 GB RAM and NVIDIA GeForce GT 730 GPU, using TensorFlow 2.4.0 [53].

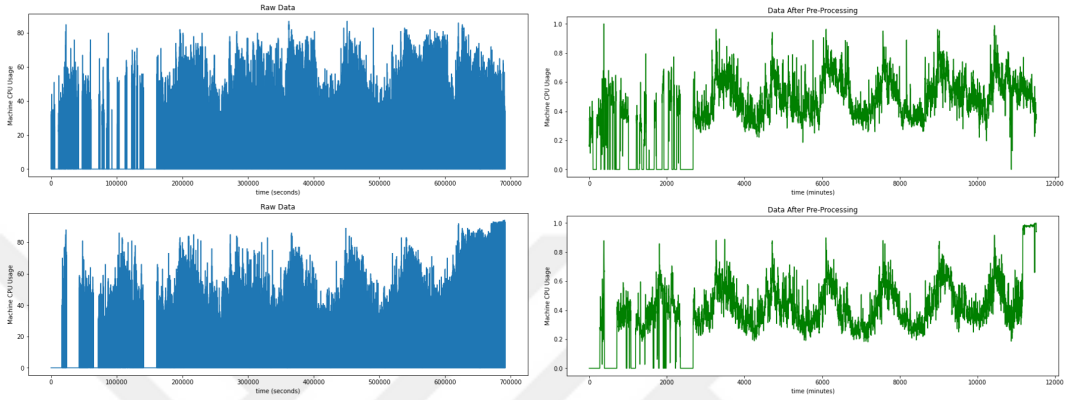


(a) Raw data from Alibaba cluster trace data set.



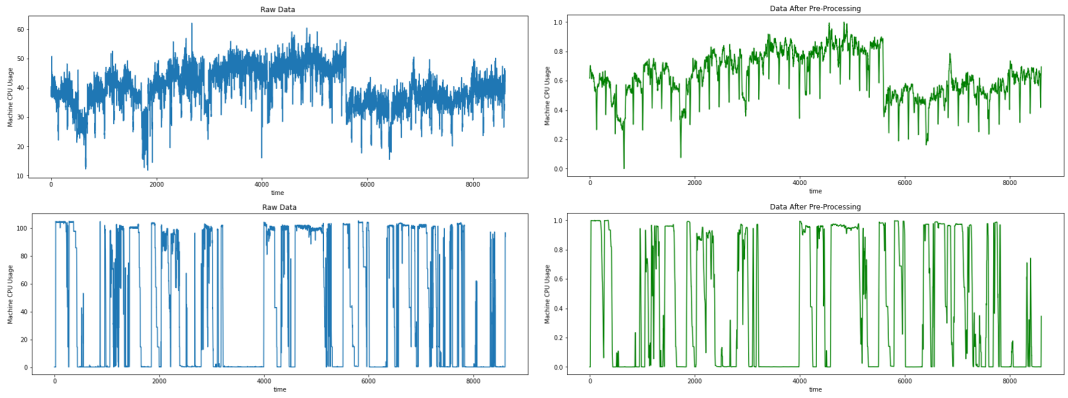
(b) Raw data from Bitbrains trace data set.

Figure 4.1: Raw data of CPU utilization for two machines from (a) Alibaba cluster trace and (b) Bitbrains trace.



(a) Raw data. (b) Pre-processed data.

Figure 4.2: CPU utilization data for two machines from Alibaba cluster trace (a) before and (b) after pre-processing.



(a) Raw data. (b) Pre-processed data.

Figure 4.3: CPU utilization data for two machines from Bitbrains trace (a) before and (b) after pre-processing.

Chapter 5

Results

This chapter presents the experimental results about the models proposed for workload prediction in cloud computing. In order to address the challenges of this problem such as determining the number of past observations and forecast horizon, various experiments are conducted on these models. The results are analyzed considering the effects on model performance.

Our performance evaluation of the deep learning-based models is based on MSE of the workload predictions.

5.1 Selection of the Number of Past Observations

The number of past observations is an important aspect of a model designed for workload prediction as the amount of historical workload data should be enough for the model to extract patterns and learn workload behaviors. In order to select the right value, we conducted experiments using various values for the number of past observations (n_{in}) while keeping the forecast horizon constant ($n_{out} = 2$). Table 5.1 shows MSE of the proposed models when $n_{out} = 2$ and $n_{in} \in \{8, 16, 32, 64\}$.

Table 5.1: MSE of the deep learning models for various levels of past observations when the forecast horizon (n_{out}) is 2. Bold MSE values show the minimum for each model. MSE is calculated on the validation set.

	Alibaba Cluster Trace Data Set					**Bitbrains Trace Data Set**				
	LSTM Network	***CNN***	***En-De LSTM Network***	***En-De LSTM Network w/ Att.***	***Compo. LSTM Autoen. Network***	***LSTM Network***	***CNN***	***En-De LSTM Network***	***En-De LSTM Network w/ Att.***	***Compo. LSTM Autoen. Network***
$n_{in} = 8$	0.003965	**0.005559**	0.003506	**0.002889**	0.003758	0.003227	**0.004249**	0.002641	0.001849	0.003131
$n_{in} = 16$	**0.003854**	0.005742	**0.003210**	0.002914	0.003401	0.003102	0.005012	**0.002529**	**0.001665**	**0.002792**
$n_{in} = 32$	0.003956	0.005744	0.003481	0.002975	**0.003393**	**0.003024**	0.004561	0.002562	0.001763	0.002956
$n_{in} = 64$	0.004005	0.006513	0.003396	0.003028	0.003595	0.003908	0.006991	0.003384	0.002168	0.003933

From this table, it can be inferred that except for CNN, the shortest historical data span leads to poor learning of the workload patterns. Although the best value for the number of past observations seems to vary depending on model type and data set, the lowest MSE is observed when it has moderate values, where $n_{in} = 16$ or $n_{in} = 32$. The tendency of CNN to use relatively shorter span can be attributed to the lower complexity of the 1D convolution operation, which reduces the model's ability to capture long-term dependencies over the workload sequence.

It is important to note that experimenting with the amount of historical workloads does not necessarily imply that a longer span of historical data would increase the error. The motivation is to find the sufficient amount of historical data to accurately predict future workloads.

Since the majority of the models show minimum error when $n_{in} = 16$, the number of lag observations (n_{in}) is set to 16 and prediction length (n_{out}) is set to 2 time steps. Under these conditions, Table 5.2 and Table 5.3 present the performance of each model on Alibaba and Bitbrains data sets, respectively. The results show that Encoder-Decoder LSTM Network with Attention outperforms other models with MSE of 0.002980 on Alibaba's test set and 0.001650 on Bitbrains's test set.

Figure 5.1 and Figure 5.2 show the workload predictions generated by the proposed deep learning models for sample machines selected from Alibaba cluster trace and Bitbrains trace data set, respectively.

Table 5.2: MSE of the proposed deep learning models on Alibaba cluster trace data set.

	Alibaba Cluster Trace Data Set		
	Training Set	***Validation Set***	***Test Set***
LSTM Network	0.003510	0.003854	0.003918
CNN	0.005254	0.005742	0.005842
Encoder-Decoder LSTM Network	0.002849	0.003210	0.003270
Encoder-Decoder LSTM Network with Attention	0.002592	0.002914	0.002980
Composite LSTM Autoencoder Network	0.003053	0.003401	0.003474

Table 5.3: MSE of the proposed deep learning models on Bitbrains trace data set.

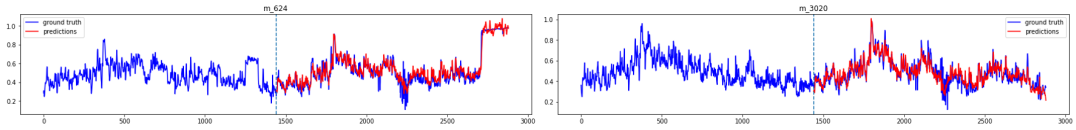
	Bitbrains Trace Data Set		
	Training Set	***Validation Set***	***Test Set***
LSTM Network	0.002860	0.003102	0.003029
CNN	0.004713	0.005012	0.004894
Encoder-Decoder LSTM Network	0.002349	0.002529	0.002504
Encoder-Decoder LSTM Network with Attention	0.001496	0.001665	0.001650
Composite LSTM Autoencoder Network	0.002542	0.002792	0.002716

5.2 The Effects of Past Observations and Forecast Horizon on Model Performance

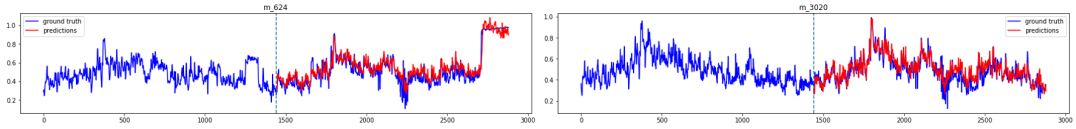
In order to get an overall evaluation of the model performance, further experiments are conducted by expanding the parameter set regarding the historical workloads and prediction length. More specifically, MSE is recorded for the deep learning models under each pair of n_{in} and n_{out}, where:

$$n_{in} \in \{8, 16, 32, 64\}$$

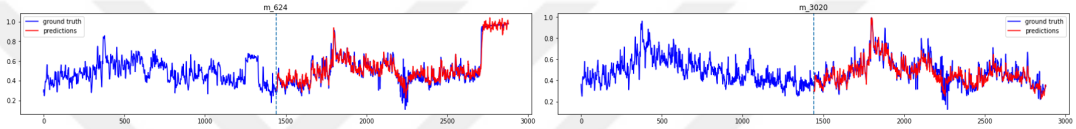
$$n_{out} \in \{2, 3, 4, 5\}$$



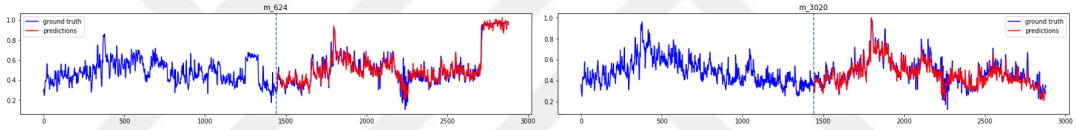
(a) LSTM Network.



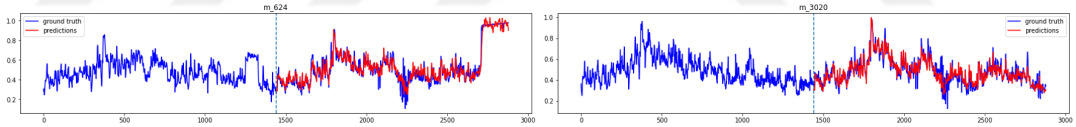
(b) CNN.



(c) Encoder-Decoder LSTM Network.



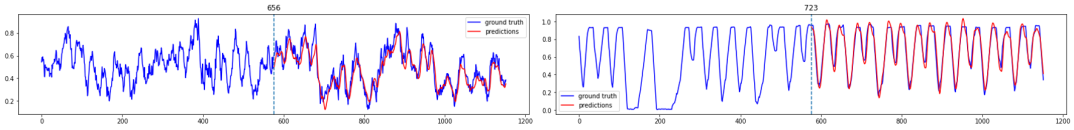
(d) Encoder-Decoder LSTM Network with Attention.



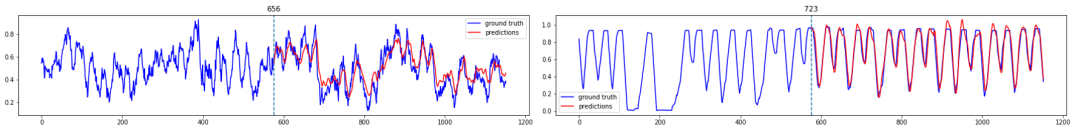
(e) Composite LSTM Autoencoder Network.

Figure 5.1: Workload predictions generated by deep learning models on Alibaba cluster trace data set.

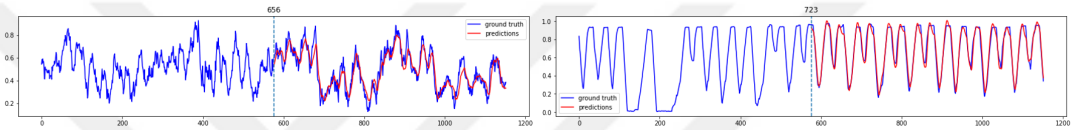
The purpose of these experiments is not only to have a wider perspective towards the capabilities of each model, but also to understand their strengths and weaknesses. The results for Alibaba and Bitbrains data sets are visualized in Figure 5.3 and Figure 5.4, respectively. As can be seen from the bar plots, regardless of the model type, the error tends to increase as the prediction length gets longer. Although there is a slight increase in the overall error for the Encoder-Decoder LSTM Network and Composite LSTM Autoencoder Network, it seems that these models can handle a relatively longer historical workload data (i.e., $n_{in} = 64$) better than LSTM Network and CNN (see Figure 5.3c and 5.3e for Alibaba; Figure 5.4c and 5.4e for Bitbrains when $n_{in} = 64$). Moreover, the trend of the error shows a dramatic increase for both LSTM Network and CNN (see Figure 5.3a and 5.3b for Alibaba; Figure 5.4a and 5.4b for Bitbrains). There are



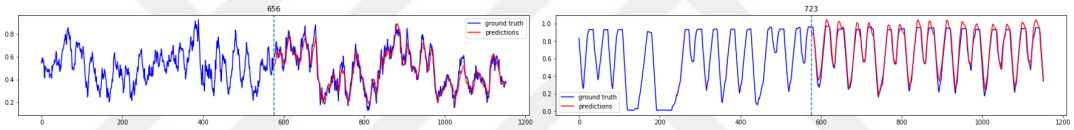
(a) LSTM Network.



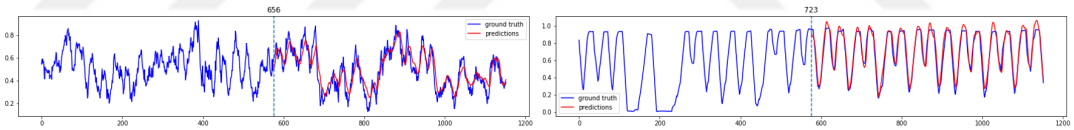
(b) CNN.



(c) Encoder-Decoder LSTM Network.



(d) Encoder-Decoder LSTM Network with Attention.



(e) Composite LSTM Autoencoder Network.

Figure 5.2: Workload predictions generated by deep learning models on Bitbrains trace data set.

also some fluctuations for CNN, which does not show a reliable pattern in case of a relatively longer forecast horizon.

Computationally expensive models, LSTM, Encoder-Decoder LSTM and Composite LSTM Autoencoder networks, appear to be considerably more accurate in predicting workloads in cloud data centers compared to CNN, which is computationally less expensive due to the 1D convolution operation. Out of these three models, Encoder-Decoder LSTM and Composite LSTM Autoencoder networks result in close MSE values as they are both able to learn the feature representations of the workloads. Encoder-Decoder LSTM Network appears to perform better than composite LSTM Autoencoder even though their architectures are very similar. This may be attributed to the fact that while Encoder-Decoder

LSTM Network specifically focus on learning the patterns for future prediction, Composite LSTM Autoencoder Network tries to simultaneously predict future steps and reconstruct the input sequence. The dual task may cause this model to fall behind Encoder-Decoder LSTM Network for this particular configuration of workload prediction problem.

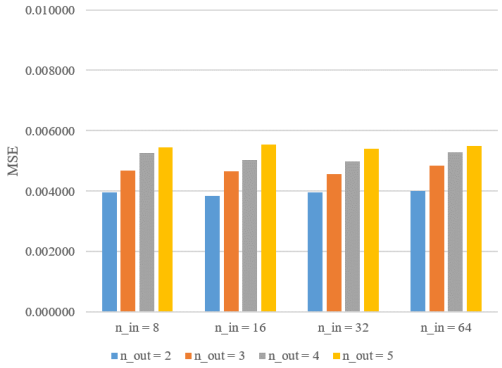
Considering the overall performance of these five deep learning-based models, the Encoder-Decoder Network with Attention appears to outperform all the other models discussed. It can be interpreted that the attention mechanism improves the performance of Encoder-Decoder LSTM Network significantly on both of the data sets.

5.3 Run Time Analysis

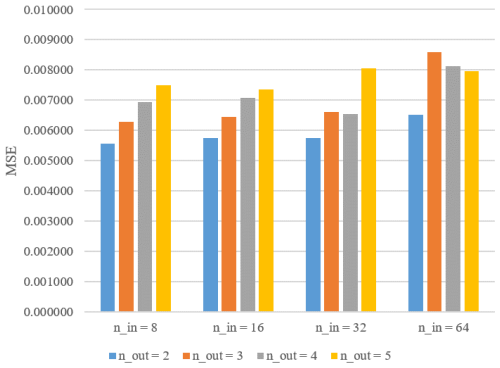
After evaluating the performance of deep learning models with respect to their accuracy on workload prediction, computational costs are analyzed. Computational cost is a significant criterion to assess if a model is both accurate and efficient in computation time.

Figure 5.5 shows the run time of each model on Alibaba and Bitbrains data sets. Run time is based on the time (in seconds) it takes for each model to process one batch from the training set. Since the dimensions of Alibaba and Bitbrains data sets are different, the run time of a model varies depending on the data set. Although both data sets are high dimensional, dimensionality of Alibaba cluster trace data set is higher than that of Bitbrains trace data set. Consequently, for all five models, run times are longer for Alibaba cluster trace data set than Bitbrains trace data set.

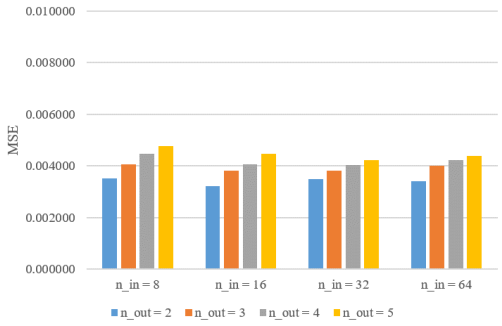
Moreover, the figure shows that CNN model is dramatically faster than all the other models due to the simplicity of the convolution operation. On the other hand, composite LSTM Autoencoder Network takes the longest time to train per step as this network not only has a relatively wider architecture in



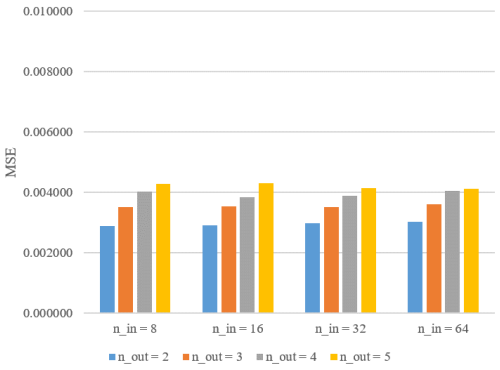
(a) LSTM Network.



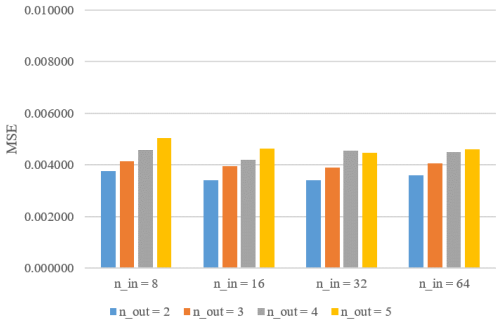
(b) CNN.



(c) Encoder-Decoder LSTM Network.

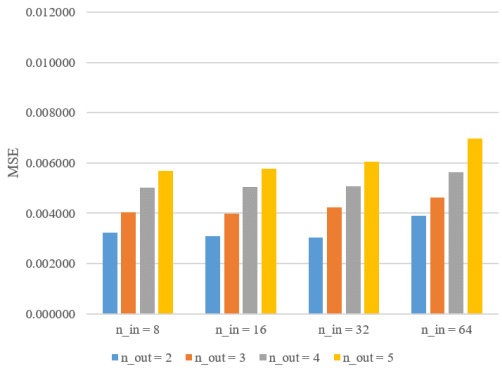


(d) Encoder-Decoder LSTM Network with Attention.

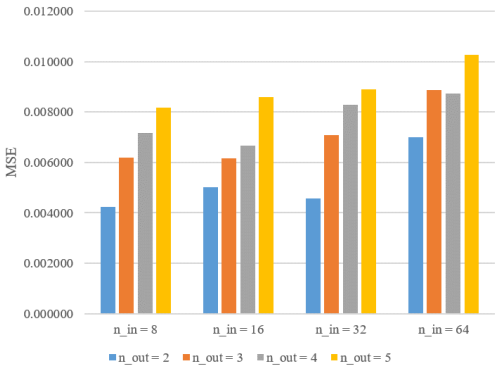


(e) Composite LSTM Autoencoder Network.

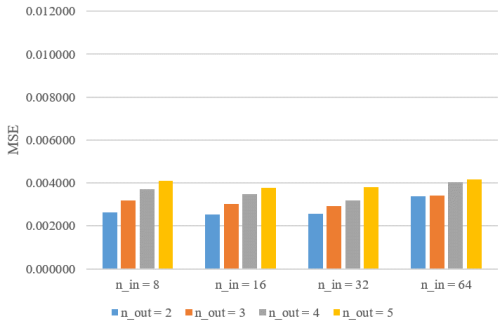
Figure 5.3: Experiments with the number of historical workloads (n_{in}) and forecast horizon (n_{out}) on Alibaba cluster trace data set. The plots show MSE of workload prediction on validation data set under each pair of (n_{in}, n_{out}).



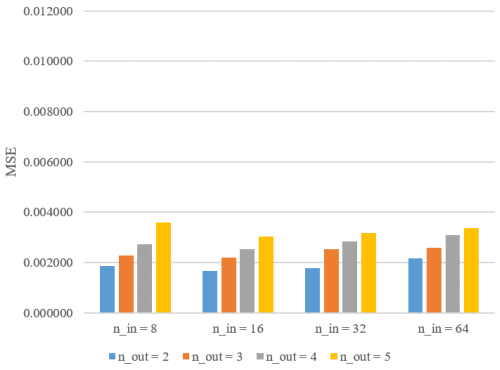
(a) LSTM Network.



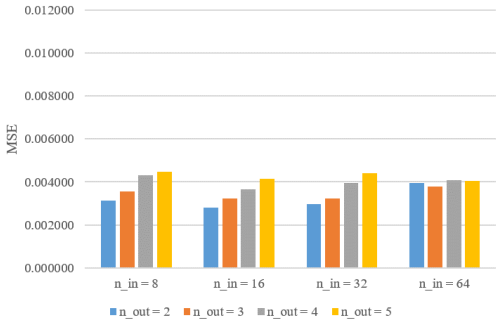
(b) CNN.



(c) Encoder-Decoder LSTM Network.



(d) Encoder-Decoder LSTM Network with Attention.



(e) Composite LSTM Autoencoder Network.

Figure 5.4: Experiments with the number of historical workloads (n_{in}) and forecast horizon (n_{out}) on Bitbrains trace data set. The plots show MSE of workload prediction on validation data set under each pair of (n_{in}, n_{out}).

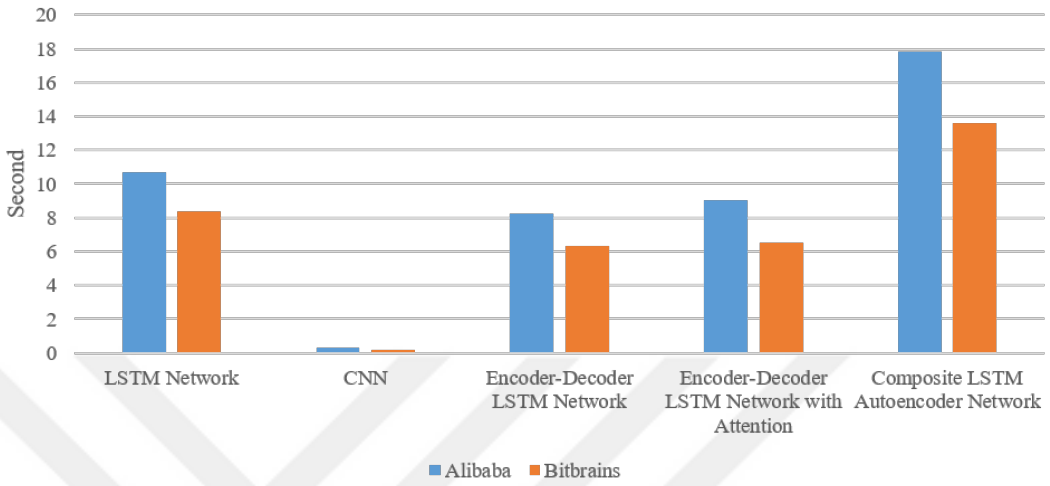


Figure 5.5: Run time of each deep learning-based model.

terms of number of hidden layers, but also deals with two tasks at the same time: reconstruction and future prediction.

Including LSTM hidden layers in the network seems to significantly increase the run time. However, encoder-decoder architectures (Encoder-Decoder LSTM Network and Encoder-Decoder LSTM Network with Attention) take shorter time than LSTM Network itself. With a deeper perspective towards the run times of encoder-decoder architectures, it can be interpreted that attention mechanism significantly reduces error by slightly increasing the run time of the model, which can be attributed to the calculation of alignment scores with *dot* method. This could be considered as a reasonable trade-off between run time and prediction accuracy.

Chapter 6

Conclusion

With the recent advances in cloud computing, service offerings provided by cloud data centers are in high demand. Along with high availability, reliability, scalability and robustness, service providers also offer low cost to their customers. Consequently, they have to follow SLAs for high quality service to their customers and utilize their cloud data center resources more effectively and efficiently to increase their profits. Therefore, a proactive resource allocation and scaling policy appears to be essential while managing data centers. Although resource scaling policies are affected by various factors such as system status, provisioning of data center resources such as CPU, RAM and disk space has become a crucial task. An efficient provisioning of resources for scaling and allocation policies can be achieved by accurate prediction of workloads. In other words, proactive policies aim at scaling the resources before the arrival of actual workloads.

Workload prediction in cloud computing is helpful in scaling and allocation of resources, therefore, reducing the number of SLA violations, failed tasks and power consumption. However, this task holds significant challenges due to the high dimensionality, variance, and complexity of the historical workloads. Moreover, prediction models are highly influenced by the number of past observations and forecast horizon. Therefore, they are not only expected to operate with a decent number of past observations so that workload patterns can be extracted,

but also handle relatively longer forecast horizons. In order to address these challenges and propose an effective workload prediction method, we present and compare 5 deep learning-based models: LSTM Network, CNN, Encoder-Decoder LSTM Network, Encoder-Decoder LSTM Network with Attention, and Composite LSTM Autoencoder Network. The performance of each models is analyzed for two real-world data sets: Alibaba cluster trace and Bitbrains trace. As a result of our comparative study, Encoder-Decoder LSTM Network with Attention is proposed as an accurate and efficient solution for workload prediction in cloud computing. A further goal is to use the output of our workload prediction model to utilize the cloud environment resources through effective and efficient resource allocation and scaling policies.

Bibliography

[1] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[2] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, pp. 843–852, PMLR, 2015.

[3] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[5] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and computer-integrated manufacturing*, vol. 28, no. 1, pp. 75–86, 2012.

[6] M. Noshy, A. Ibrahim, and H. A. Ali, "Optimization of live virtual machine migration in cloud computing: A survey and future directions," *Journal of Network and Computer Applications*, vol. 110, pp. 1–10, 2018.

[7] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.

[8] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 923–934, 2019.

[9] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing*, pp. 1–26, 2019.

[10] Z. Huang, J. Peng, H. Lian, J. Guo, and W. Qiu, "Deep recurrent model for server load and performance prediction in data center," *Complexity*, vol. 2017, 2017.

[11] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "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)*, pp. 104–109, IEEE, 2016.

[12] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proceedings of the 9th international conference on Autonomic computing*, pp. 145–154, 2012.

[13] Y. Guo and W. Yao, "Applying gated recurrent units approaches for workload prediction," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, IEEE, 2018.

[14] 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)*, pp. 67–72, IEEE, 2017.

[15] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, pp. 25–32, IEEE, 2015.

[16] Y. Yu, V. Jindal, F. Bastani, F. Li, and I.-L. Yen, "Improving the smartness of cloud management via machine learning based workload prediction," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 38–44, IEEE, 2018.

[17] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium*, pp. 1287–1294, IEEE, 2012.

[18] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 1–12, 2019.

[19] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host cpu utilization in the cloud using evolutionary neural networks," *Future Generation Computer Systems*, vol. 86, pp. 162–173, 2018.

[20] J. Cao, J. Fu, M. Li, and J. Chen, "Cpu load prediction for cloud environment based on a dynamic ensemble model," *Software: Practice and Experience*, vol. 44, no. 7, pp. 793–804, 2014.

[21] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, IEEE, 2017.

[22] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.

[23] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, and S. Du, "Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3037–3053, 2015.

[24] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, vol. 1, 2011.

[25] H. Liu, "A measurement study of server utilization in public clouds," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 435–442, IEEE, 2011.

[26] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload prediction for cloud computing elasticity mechanism," in *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 244–249, IEEE, 2016.

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

[28] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE transactions on cloud computing*, vol. 3, no. 4, pp. 449–458, 2014.

[29] B. Liu, Y. Lin, and Y. Chen, "Quantitative workload analysis and prediction using google cluster traces," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 935–940, IEEE, 2016.

[30] W. Iqbal, J. L. Berral, A. Erradi, D. Carrera, *et al.*, "Adaptive prediction models for data center resources utilization estimation," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1681–1693, 2019.

[31] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Computing*, vol. 21, no. 3, pp. 1581–1593, 2018.

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

[33] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–18, 2019.

[34] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2015.

[35] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.

[36] R. Assaf and A. Schumann, "Explainable deep neural networks for multivariate time series predictions.," in *IJCAI*, pp. 6488–6490, 2019.

[37] R. Fukuoka, H. Suzuki, T. Kitajima, A. Kuwahara, and T. Yasuno, "Wind speed prediction model using lstm and 1d-cnn," *Journal of Signal Processing*, vol. 22, no. 4, pp. 207–210, 2018.

[38] A. Haidar and B. Verma, "Monthly rainfall forecasting using one-dimensional deep convolutional neural network," *IEEE Access*, vol. 6, pp. 69053–69063, 2018.

[39] R. S. Srinivasamurthy, *Understanding 1D Convolutional Neural Networks Using Multiclass Time-Varying Signals*. PhD thesis, Clemson University, 2018.

[40] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Multivariate time series forecasting via attention-based encoder–decoder framework," *Neurocomputing*, vol. 388, pp. 269–279, 2020.

[41] I.-F. Kao, Y. Zhou, L.-C. Chang, and F.-J. Chang, "Exploring a long short-term memory based encoder-decoder framework for multi-step-ahead flood forecasting," *Journal of Hydrology*, vol. 583, p. 124631, 2020.

[42] L. Roeder, "Netron, Visualizer for neural network, deep learning, and machine learning models," 12 2017.

[43] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[44] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," *Advances in neural information processing systems*, vol. 26, 2013.

[45] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[46] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[47] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[48] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, IEEE, 2019.

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

[50] A. Kwan, *Predicting Fine-grained Cloud Resource Usage with Online and Offline Learning*. PhD thesis, University of Toronto (Canada), 2020.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[52] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international*

conference on artificial intelligence and statistics, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[53] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.