QATAR UNIVERSITY

COLLEGE OF ENGINEERING

DEEP LEARNING BASED APPROACH FOR PREDICTION OF CLOUD RESOURCE NEEDS

BY

ANWAR AL-QUTAN

A Project Submitted to

the Faculty of the College of

Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Masters of Science in Computing

June 2018

COMMITTEE PAGE

The members of the Committee approve the Project of Anwar Al-Qutan

defended on 27/05/2018.

_____

Dr. Abdelkarim Erradi
Thesis/Dissertation Supervisor

_____

Dr. Osama Halabi
Committee Chair

_____

Prof. Ali Jaoua
Committee Member

_____

Dr. Khaled Bashir Shaban
Committee Member

ABSTRACT

AL-QUTAN, ANWAR, MEFLEH., Masters : June : [2018], Masters of Science in Computing

Title: Deep Learning Based Approach for Prediction of Cloud Resource Needs

Supervisor of Project: Abdelkarim Erradi.

Cloud computing allows scaling applications to serve dynamic and time-varying workloads and to avoid application performance degradation, while keeping low provisioning costs. But, resource demand of applications need to be determined beforehand. Therefore, accurate prediction of cloud resource needs is critical by enabling proactive scaling to efficiently manage cloud resources and to reduce the operational cost. Most of the exiting resource prediction approaches are based on the statistical analysis that employ shallow structure. As a result, the prediction model has poor ability to capture the intrinsic features in the workload data.

Deep learning has emerged as an alternative approach that promise to produce more accurate prediction. This project designed, implemented and evaluated a deep learning based approach for prediction of cloud resources using Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP). Moreover, a statistical prediction model Autoregressive Integrated Moving Average (ARIMA) is developed and evaluated. Expensive experimental studies were performed to evaluate the accuracy of deep learning prediction models compared to traditional ARIMA approach. The result of the experiments shows that the prediction accuracy of LSTM, MLP and ARIMA models depend on the pattern of the incoming workload. Specifically, the result shows that LSTM model outperforms other prediction models for periodic workload patterns, while ARIMA has better prediction accuracy for growing and unpredicted workload patterns.

DEDICATION

To My Brother Naser,

Who always inspires me, I'm forever proud of you

To my children Abdullah, Dana, Qais and Mohammad,

Without you this work would have been completed one year earlier. I am truly thankful for having you in my life.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Dr. Abdelkarim Erradi, for his sustained support, guidance, immense knowledge and motivation throughout my project journey. It was an honor to work with him.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

- VM: Virtual Machine
- PM: Physical Machine
- SLA: Service Level Agreement
- IaaS: Infrastructure as a Service
- PaaS: Platform as a Service
- SaaS: Software as a Service
- ANN: Artificial Neural Network
- MLP: Multi-Layer Perceptron
- RNN: Recurrent Neural Network
- LSTM: Long Short-Term Memory
- MAE: Mean Absolute Error
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- ARIMA: Autoregressive Integrated Moving Average
- DBN: Deep Belief Network
- RBM: Restricted Boltzmann Machine
- RL: Regression Layer
- SVM: Support Vector Machine

CHAPTER 1: INTRODUCTION

In cloud computing, resource management is the process of allocating resources, such as computing, storage or networking to a set of applications in a way that meets the performance objective of the applications, cloud providers and cloud users. Resource management is considered a hard problem due to many reasons including heterogeneity of resource types, unpredictability of the load, large scale of today's data center and the variety of objectives of different cloud actors [1].

Resource provisioning is one of the most important problems of resource management for Infrastructure as a Service (IaaS) cloud model. The main two resource provisioning mechanisms are reservation and on-demand plans. In reservation plan the client pays an in-advance fee and uses the resources when required. On other hand, on-demand plan allows the client to pay for a new instance whenever required without any long-term commitment. The price of the reserved instance is cheaper than on-demand instance thus enabling significant savings when used. However, it suffers from two main problems. First, is the risk of over-provisioning of reserved instances which result in unnecessary monetary loss when some of the reserved resources are underutilized. Second, is the risk of under-provisioning where the reserved resources are unable to meet the actual demand, causing poor application performance leading to Service Level Agreement (SLA) violation.

The availability of reserved instances is guaranteed by the cloud providers at a pre-determined price, but the availability and price of on-demand instances may vary significantly. To reduce the resource provisioning cost, we need an optimal in-advance reservation of resources. However, this is not an easy task due to demand uncertainties

from cloud consumer side and price uncertainties from cloud provider side [1] Successful planning of in-advance reservation of resources requires a robust and long-term prediction mechanism to accurately predict future resource needs. Moreover, the speed of response to the workload changes to attain the intended level of performance is a critical for cloud elasticity [12]. Hence, to enable elasticity of resources allocation an auto-scaling system is required to adjust the allocated resources for the workload generated by application. Auto-scalers can be categorized according to their anticipation capabilities into reactive and proactive [3]. Reactive when the system reacts in response to the detected changes in demand. Whereas proactive reacts in response to predicted future demand. Hence, proactive auto-scaling requires accurate predicting of future resource demands.

The salient issues faced by cloud computing regarding service provisioning is the critical need to accurately predict workloads of virtual machines (VMs). This will be beneficial in terms of cost-effective resources allocation as the consumer will only need to pay for what is needed and used.

Time series analysis might be used to find the repeating pattern in input workload to predict future values [14]. Hence it can possibly be used for prediction of cloud resources. Machine learning-based techniques, such as regression and neural network are considered one group of time series analysis that focus on direct future prediction [3].

1.1.Problem Statement

In cloud computing, virtual machine (VM) workload impacts resource provisioning. Most auto scaling approaches are reactive as they add/remove Virtual Machines (VMs) when resources utilization cross certain threshold. Realizing the full potential of the cloud using proactive auto-scaling remains challenging particularly due to the need to accurately estimate the application resource requirements for time-varying workload. Therefore, accurate prediction of virtual machine resource needs is very critical to manage the cloud resources efficiently and reduce the cost of resources. Most of the exiting resource prediction approaches based on the statistical analysis that employ shallow structure or based on neural network with one hidden layer. In consequence, the prediction model has poor ability to find the intrinsic features in the workload data, due to the randomness and nonlinear nature of the workload data [4].

Deep learning has been found to be appropriate for big data analysis with many successful applications to computer fields, such as computer vision and pattern recognition. They have been shown to produce better results on various tasks [5]. Based on that, deep learning emerges as a candidate approach that promises to produce more accurate prediction. Therefore, this research, will use deep learning to build a resource prediction model to address the problem of cloud resource needs.

The specific objectives of this study are the following:

- Design and implement a deep learning model to forecast future cloud resource needs based on historical data.

- Compare the prediction accuracy of deep learning prediction model with the traditional approaches.

1.2. Proposed Approach

This project applies and compares various approaches for prediction of cloud resource needs. A Feed-Forward Multilayer Perceptron (MLP) will be developed, then a Long Short-Term Memory (LSTM) Recurrent Neural Network will be built as it is specifically designed for sequence prediction. Also, an Autoregressive Integrated Moving Average (ARIMA) model will be built, which is a popular and commonly used statistical model for time series forecasting. All the thereafter created prediction models will be trained using the same dataset and then evaluated. After that, the results are compared.

The implementation is based on the best of breed Python deep learning library called Keras [1]that uses TensorFlow [2] as a backend [6].

1.3. Project Contributions

Following is a list of the main contributions of this work:

- Conduct a literature review of traditional and deep learning methods used in time series prediction in the context of cloud resource predictions.
- Design and implement MLP and LSTM for resource prediction for facilitating proactive auto scaling of cloud resources.
- Evaluate and compare MLP, LSTM and ARIMA prediction accuracy to predict cloud resource needs for the different workload patterns.

[1] https://keras.io/
[2] https://www.tensorflow.org/

- Analyzing the impact of using multiple input variables (multivariate features) on MLP and LSTM prediction accuracy.

The rest of the report is organized as follows. Chapter 2 provides a background and surveys the literature for time series prediction using deep learning techniques and other traditional methods. Chapter 3, presents the dataset, the data exploration and preparation for building the deep learning models. The methodology used for designing and building the deep learning prediction models is explained in chapter 4. Chapter 5 provides the evaluation and discussion. Chapter 6 concludes and suggests future works.

CHAPTER 2: BACKGROUND AND RELATED WORK

In this chapter the background and basic concepts used in this research project and related work are introduced. Section 2.1 is an overview of resource management in cloud computing and time series analysis. Section 2.2 provides the literature review.

2.1. Background

Cloud computing is an Internet-based computing that involves the delivery of hosted services over the Internet using pay-as-you-go model.

2.1.1. Resource Management in Clouds

Resource management is the process of allocating resources to a group of applications in a way that meets the performance objective of the applications, of the cloud providers and of the cloud users. In general, cloud environment has three different actors: a Cloud Provider, a Cloud User and an End User. They have different objectives and play different roles. The objectives of the cloud provider are to focus on efficient and effective use of resources within the constraints of SLAs with cloud user. The cloud user cares more about the application performance, availability and cost-effective scaling. From the cloud resource management perspective, the cloud provider manages the resources by allocating them in a way that meets SLAs. The cloud user uses the resources to host applications. The end user generates the workloads which are processed by cloud resources.

2.1.2. Auto-scaling Techniques

Elastic applications dynamically provision/release computing resources to meet a varying workload. To enable the elasticity, an auto-scaling system is needed for automatic resources adjustments in order to handle the applications' workload. The auto-scalers make dynamic scaling decisions without human intervention.

Scalability falls into two categories: (a) horizontal scaling that involves adding/removing nodes, (b) vertical scaling that involves adding/removing resources to a single node in the system. Horizontal scaling is offered by many cloud providers. The auto-scaling process has the following four phases (MAPE loop):

- Monitoring phase: involves gathering performance metrics of the system and the state of its application at different time granularity.
- Analysis phase: obtaining the current system utilization and/or predictions of future demand.
- Planning phase: in this phase resource modification action (e.g. add/remove VM, upgrade memory) can be planned in a way that retains a good balance between cost and SLA compliance.
- Execution phase: execute the action.

Auto-scalers can be reactive or proactive. Reactive where the system reacts in response to the detected changes in demand. The reactive auto-scaling may fail to respond to rapid changes in the workload, especially in cases of sudden spike (a new virtual machine usually takes 5 to 10 minutes to be operational). Due to time needed to obtain and configure new resources, proactive auto-scaling is more promising technique

as it reacts in response to predicted future demand.

According to [3] auto-scalers can also be categorized according to the underlying technique used for implementation of auto-scaler into:

- **Threshold-based rules:** The scaling decision in the threshold-based rules are based on some performance metrics (such as average of input request, average of response time and average CPU utilization) and pre-defined threshold. This technique is considered to be simple, but suffers from the difficulty of choosing the right performance metrics.

- **Reinforcement learning:** This method learns from past experience (trial and error method) the best obtainable option taken in each particular state. The auto-scaler gets a feedback from the system after executing every action that shows the goodness of the performed action, therefore the auto-scaler has always a tendency to execute the best actions. The main drawback of this technique is the long learning phases and bad initial performance.

- **Queuing theory:** Queuing theory relies on modeling the system for determining its future resource needs. It involves finding relationships between arriving/leaving jobs from the system. Elastic application can be formed using a simple queuing model in which the load balancer is represented by single queue that distribute the request among number of VMs. Simulation is often used to solve the queuing models. Some of the performance metric used average waiting time in a queue and average response time. Limitation of the queuing theory model is non-flexibility and the parameters and metrics always need to be recomputed.

- **Control theory:** Control theory requires creating a model of application. It involves defining a controller that adjusts the needed resources for the application demand. Different types of control system exist, open-loop controller, which uses the current state data in computing the input to the system without using feedback to check if the goal is achieved. Feedback controller observes the system output and corrects any differences from the target value. Finally, feed-forward controller which predicts and reacts before the errors happen. Feedback controllers are the most used control system. They adjust the input variables according to the target value in the output variable. However, the problem of creating a dependable performance model that maps input to output variables remains challenging.

- **Time series analysis:** In time series analysis technique, a certain performance metric sampled periodically at fixed intervals for prediction of future values. Time series analysis can be used to find repeating patterns in the input workload to predict future values.

2.1.3. Time Series Analysis and Forecasting

Time series is widely used in various domains to represent the change of measurement over time. The time series are a sequence of data points that are measured at equally spaced time intervals. Time series analysis uses a variety of methods for patterns detection and for future values prediction

The main methods used for prediction include:

- **Moving Average (MA) methods:**

The method is used to remove the noises or to make predictions. The prediction formula

calculates the weighted average of the last history window of the successive values. Similarly, there are different variations, such as simple moving average that assigns equal weight to all observations. It is only useful for forecasting when there is no trend. Weighted moving average gives a different weight to each observation value. In which more weight is given to the recent observations. Moving averages methods are still not able to handle significant trends when forecasting.

- **Exponential smoothing (ES)**

ES is similar to MA, it is used to compute the weighted average of past observations. However, it considers the past history of the time series. It earmarks exponentially decreasing weights over time. There are various versions of ES, such as S*ingle ES, Double ES and Triple ES.* Single (i.e. simple) exponential smoothing is insufficient in case of time series with changing trends. However, when there is a linear trend, double ES is a good choice.

- **Auto-regression of order p, AR(p)**

The formula to make prediction is defined according to a linear weighted sum of past values. The main idea is to obtain the best values for the weights or auto-regression coefficients. There are different techniques used to compute the AR coefficients. The most common ones are those based on the calculation of auto-correlation coefficient and on Yule-Walker equations [3].

- **Auto-Regressive Moving Average (ARMA)**

ARMA is a combination of both auto-regression and moving average. It is suitable for stationary time series which means the time series have no trend or

seasonality. However, the Auto-regression Integrated Moving Average (ARIMA) which is an extension to ARMA model may be used for non-stationary time series.

- **Machine learning-based techniques**

The most popular machine learning-based techniques are regression and neural networks. Regression is a widely used technique that based on statistical model for finding out an estimation about the relationships between variables. A neural network is connected group of nodes (neurons) that consist of many layers, which include input layer, output layer, and hidden layer/s in between. According to [3], the neural network performs better accurate results than MA and simple ES methods.

The other group of time series analysis focuses on identifying patterns of time series which follows and then uses these patterns for predicting future demand. Normally, it has four classes of components, (a) Trend, (b) Seasonality, (c) Cyclical and (d) Randomness. The following techniques could be used to determine the repeating patterns:

- Pattern matching technique.
- Fast Fourier Transform (FFT).
- Auto-correlation.

Generally, time series analysis is considered to be the key took for proactive auto-scaling. However, its main drawback is that the prediction accuracy depends on the application workload.

2.1.4. Time Series Forecasting with Deep Learning

Deep learning is "a class of machine learning techniques that exploit many layers of non-linear information processing for supervised, or unsupervised feature extraction, and for pattern analysis and classification" [26].

Time series forecasting is different from other regression predictive modeling where the sequence adds an order dependence on the observation that must be well-preserved during training the model and making predictions. Different deep learning architectures that could be used for time series forecasting were found in the literature. The two most common among them are Multilayer Perceptron and Long Short-Term Memory.

2.1.4.1. Multilayer Perceptron

One of the most widely used machine learning techniques for prediction is Artificial Neural Networks (ANN) or Multilayer Perceptron (MLP). The power of neural networks is their ability to learn the representation of training data and relate this representation to output variable that we need to predict. The predictive capability of ANN comes from hierarchical structure of the networks, which allows representing features at different scales. The basic building block of neural networks, are neurons, which are simple computational models that have weighted input and produce output value using an activation function [11]. The activation function is a simple mapping between summed weighted input and the output of the neuron. In Multilayer Perceptron, neurons are grouped into layers, and these layers grouped into networks called network topology.

Neural networks need to be trained on the dataset to build a model that we can use to make predictions. The common training algorithm for artificial neural networks is stochastic gradient descent. This is where each row of data is exposed to the network as input and processed upwards activation function to produce an output. This phase is called forward pass. The second phase is the back propagation pass where the output of the network is compared to the expected value that we already have in the dataset and calculate the error. which then propagated back through the network and the weights are updated according to the amount of error they participated in.

MLPs are useful for many machine learning problems due to their ability to solve problems stochastically. However, they have some limitation with respect to the problems that involve sequence prediction. In the next section a special type of neural networks that is designed for sequence problems will be introduced.

2.1.4.2. Long Short-Term Memory

In general, Recurrent Neural Networks (RNNs) are a special type of neural network designed for sequence problems. RNN can be thought as adding loops the architecture of the standard feedforward Multilayer Perceptron network. Another way to think about RNNs is the addition of memory to the network that allow it to store and remember information, and utilize the ordered nature of observations within input sequences.

The Back-propagation algorithm is used to train feedforward neural networks breakdown in RNN due to loop connections. As a result, a new technique is used for training RNN called Back-propagation Through Time (BPTT) where the structure of the network is unrolled (copies of neurons are created) into a full network. Figure 1 shows a

typical RNN (left) being unrolled (right). This chunk of NN (A) looks at some input x_t and outputs a value h_t. A loop in RNN permit information to be passed from one step to the next.

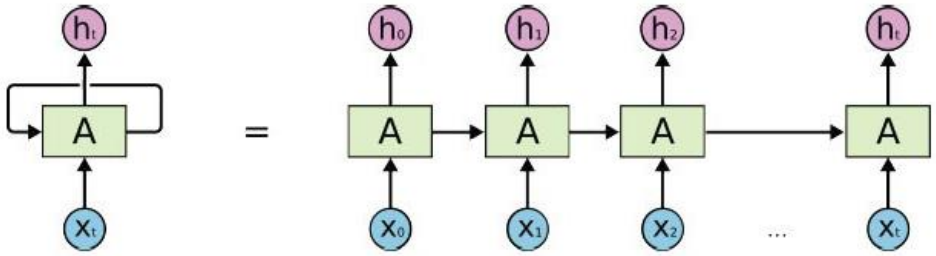

Figure 1. Typical RNN and the unrolling in time[3]

When Back-propagation used in very deep learning and in unrolled RNN, the gradients which are generated to update the weights can become unstable, they might become very small which results in vanishing gradient problem, or a very large which also results in exploding gradients [13]. This problem is solved in deep Multilayer Perceptron network by using the Rectifier function, or unsupervised pre-training of layers. In RNN it is solved or reduced by using a new type of architecture called Long Short-Term Memory.

The Long Short-Term Memory (LSTM) is a kind of Recurrent Neural Network (RNN) designed to address the problem of vanishing gradients. The LSTM has memory blocks instead of neurons that can be connected into layers. Each block contains three main gates: a forget gate, an input gate, and an output gate. Those gates manage each

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

block's state and output by controlling h information to be discarded, updated and to be outputted. LSTM has the capability to solve many time series tasks which are unsolvable by MLP using fixed size window method [28]. Moreover, it has the capability also to solve complex "long-time-lag" tasks that have never been solved by previous RRN algorithms [29]. Figure 2 below shows LSTM cell structure that contains four NN layer (button) compared to a standard RNN that contains a single layer (top).

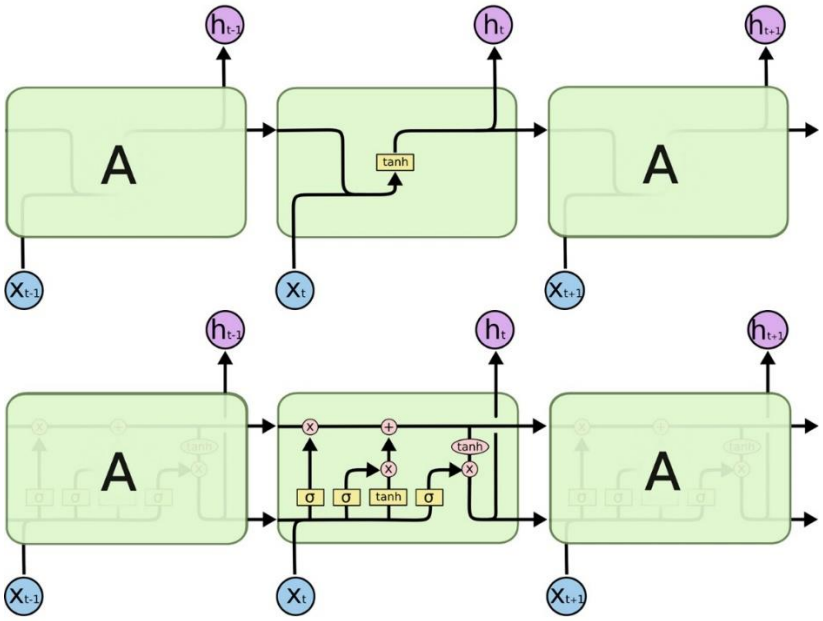

Figure 2. Unrolled RNN cell structure (top) vs. LSTM cell structure (bottom)[4]

[4] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

2.2. Related Work

The goal of the related work is to review the existing approaches in the literature that address time series prediction using traditional and deep learning approaches in the context of cloud computing. Prediction of cloud resources usage would contribute to the efficiency of cloud auto scaling systems. Most techniques found in literature focus on the prediction of workload, performance and SLA metrics. According to survey [12] the notion workload is interpreted in different ways in literature:

- Application workload is equal to the number of requests received by the application.
- The future demand of VMs. In this case, the future demand of resources of VMs are forecasted.
- Resources utilization in terms of CPU and memory.

Many methods in the literatures proposed focusing on the resources utilization. These methods forecast the usage, the load or the utilization of resources. CPU and memory are the most considered resources in the literature.

Several approaches have been proposed in order to predict VM workload, for example, Qiu et al. [4] proposed VM workload prediction (CPU utilization) based on deep learning. The prediction model was designed with a Deep Belief Network (DBN) composed of multiple-layered restricted Boltzmann machines (RBMs) and a Regression Layer (RL). The DBN was used for features extraction from all VMs workload data and the regression layer was used for workload prediction. They used CPU utilization data of all VMs of several pervious time steps as input to the proposed model and forecast one-

time step in the future of CPU utilization for one single VM. They compared their approach with exponentially weighted moving average method, simple neural networks, multilayer neural networks and ARIMA model. Their results showed that the proposed approach has better workload prediction performance than the other used workload prediction approaches.

Jheng et al. [22] proposed a workload prediction method using Grey Forecasting model to predict the workload of VMs. Specifically, they predicted the average utilization of resources CPU, Memory and RAM of physical machines. Their proposed approach uses fewer training to predict the workload and it reduces the power consumption. In [23] the authors used the following time series prediction methods: moving average, Auto-regression, ANN, Support Vector Machine (SVM) and Gene Expression Programming to forecast the future demand of VMs and the capacity planning. They proposed a novel cost-sensitive measure called Cloud Prediction Cost (CPC) to guide the prediction procedure.

Bankole et al. [24] developed three machine learning forecasting models for a web server and database performance benchmark (TPC-W) web application using Support Vector Regression, Neural Network and Linear Regression. The developed forecasting models are used to predict the CPU utilization of VMs and SLA metrics (response time and throughput). The results showed that SVM outperforms NN and LR. In [25] the auhors used Autoregressive Integrated Moving Average (ARIMA) model to propose workload prediction module for dynamic provisioning of resources for SaaS providers.

Deep learning has been successfully applied for time series prediction problems in other domains. For example, Hernández et al. [19] proposed a deep learning model based on Auto-encoder and Multilayer perceptron to forecast the daily accumulated rainfall. The authors used Auto-encoder for feature selection and MLP for making predictions. The results showed that their proposed model performs better than other approaches. The authors in [20] were able to predict heavy rainfall events for 2 days earlier of the event (6 to 48 hours before the event) with great precision. The authors used deep learning with stacked Auto-encoder for features learning, then they used them to predict extreme rainfall. Huang et al. [21] propsed a deep learning architecture for traffic flow prediction. The proposed architecture composed of Deep Belief Network (DBN) for feature learning and regression layer (RL) used for supervised training, they concluded that their approch outperfom state-of-the-art methods with 3% improvement.

The unique approach of this study is to apply Long Short Term Memory (LSTM) Recurrent Neural Network deep learning architecture for the first time in cloud resource prediction, though it had been used in other fields. Most of the techniques used in cloud for resource needs prediction opt to other resource prediction approaches. But, this study focused the research on usage of LSTM in this area counting on its suitability for reaching accurate results in cloud resource needs prediction. Moreover, this work studies the impact of selecting multiple input features on the prediction accuracy of the deep learning models.

CHAPTER 3: DATA COLLECTION AND ANALYSIS

Section 3.1 describes the dataset used for the experiments. Section 3.2 explores the dataset in order to understand it better. Section 3.3 describes the data preprocessing phases.

3.1. Data Collection

A long-term and large-scale workload traces from a distributed cloud datacenter (Bitbrains) are the dataset used in this work. The dataset contains performance metrics of 1,750 VMs. The traces include information about CPU, memory, disk I/O and network I/O, for both requested and used resources. The VMs host business-critical applications. The dataset is collected and analyzed by Shen et al. [9] and made publicly available at the Grid Workloads Archive [10]. The format of the dataset is a row-based, sampled every 5 minutes. Each row represents an observation of the performance metrics.

The dataset is very large in term of size and it has workload data for 1750 VMs and most of them contains thousands of observations. For evaluation with sensible time, we need to make a reduced sample to work with. In fact, the dataset should be small enough, so that we can build deep learning models and test them with a reasonable time and computations. Later, well performing models can be always scaled up and trained on all dataset. For the evaluation purpose, 15 samples of the dataset that represent three different workload patterns were chosen. Authors in [17,18,26] categorize workload patterns in cloud computing environment into the following three different workload patterns:

1. Periodic workload which represents workloads with cyclical (seasonal) change.
2. Growing workload which represents workload with growing trend.
3. Unpredicted workload which represents fluctuating workload.

In this project dataset samples are selected to represent these three workload patterns only. In the following section we will explore these datasets. Further, we describe the data preparation performed to prepare the input to the deep learning CPU usage prediction models designed in chapter 4. In fact, the problem can be framed to forecast other resource metric, such as memory usage. Figures 3,4 and 5 show the univariate plots of the target attribute (CPU usage) for three VMs, each with different workload patterns.

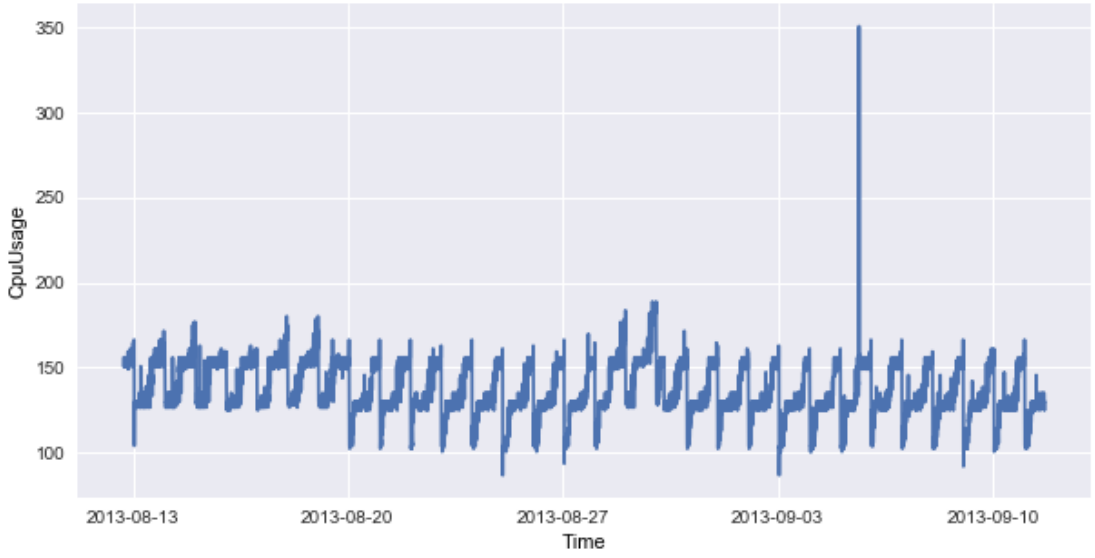

Figure 3. CPU usage over time for VM_1 having a Periodic Workload

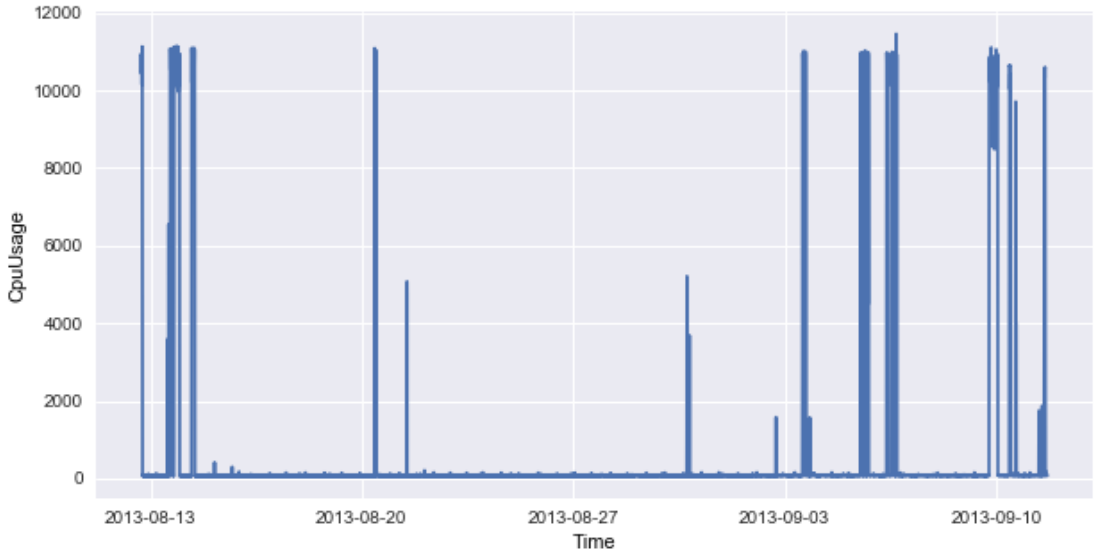

Figure 4. CPU usage over time for VM_7 having an unpredicted workload

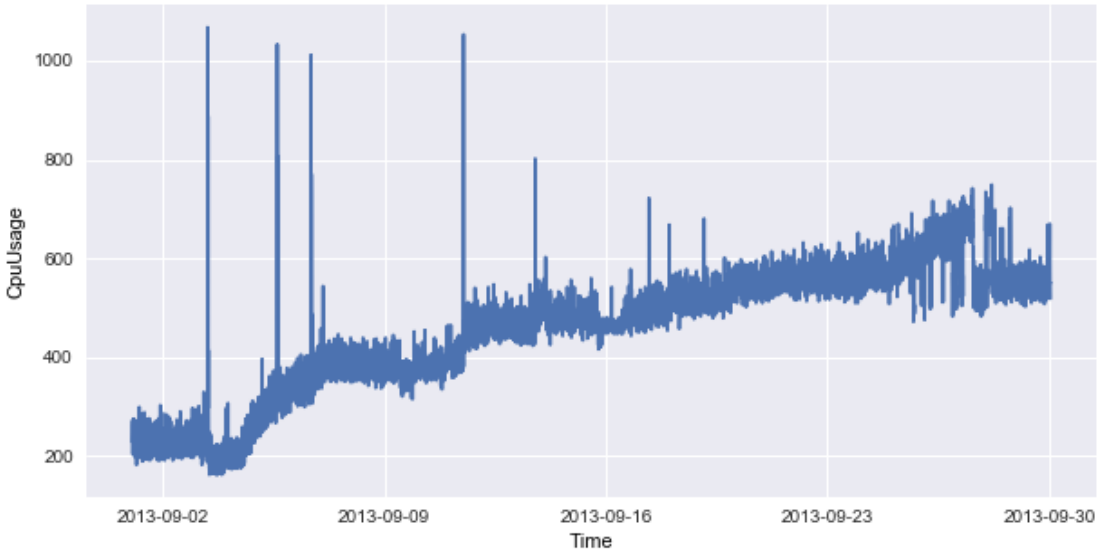

Figure 5. CPU usage over time for VM_{12} having a growing workload

3.2. Data Exploration

In machine learning, problems are solved by learning from data. Hence, firstly there is a need to prepare and understand the dataset well before starting working on the deep learning models. The dataset is explored using the following Python libraries:

1. NumPy[5]: efficient computation of multi-dimensional arrays
2. Pandas[6]: fast and flexible data structures for data analysis
3. Matplotlib[7]: 2D plotting library
4. Seaborn [8] : additional plot types, stylish and readable plots

Pandas are used to read the dataset into a data frame. Then, statistics and charts are generated to better understand the dataset.

3.2.1. Data summary

Since the type of each attribute is important for building an artificial neural network, we reviewed the data type of each attribute. The shape and the size (number of features and instances) are also reviewed. Fortunately, it was found that all of the input and output attributes are numerical. This means that there is no need for any type of conversion to integers or floating-point values required for a neural network. Table (1) shows a sample of the dataset for VM_1.

[5] http://www.numpy.org/
[6] https://pandas.pydata.org/
[7] https://matplotlib.org/
[8] https://seaborn.pydata.org/

Table 1

Sample of the VM_1 data

CpuCapacity	CpuUsage	CpuUsagePer	MemUsage	DiskWrite	NetTransmitted
10400	150.8	1.45	190141.6	3.867	5
10400	156	1.5	268435.2	4.667	5
10400	156	1.5	246065.6	4.2	5.067
10400	156	1.5	536870.4	4.4	5
10400	156	1.5	290804.8	4.333	4.933

3.2.2. Distributions of each attribute

Descriptive statistics shows how values of each attribute are distributed. This gives an insight into the shape of each attribute. The basic statistical properties that have been reviewed are: count, mean, standard deviation, minimum value, 25^{th} percentile, median, 75^{th} percentile, maximum value. Table (2) shows the distribution summary of each attribute for the first VM_1 workload data. It can be seen that VM_1 has 8621 observations sampled every 5 minutes (i.e. one month) and 137.29 is the average CPU usage.

Table 2

Summary of attributes distribution

	CpuUsage	CpuUsagePer	MemUsage	DiskWrite	NetTransmitted
count	8621	8621	8621	8621	8621
mean	137.29	1.32	359483.625	6.886	4.972
std	15.892	0.153	117971.742	8.888	0.082
min	86.667	0.833	22369.6	3.133	4.467
25%	130	1.25	279620	4.267	4.933
50%	130	1.25	346728.812	4.4	5
75%	156	1.5	436207.188	4.533	5
max	351	3.375	950708	527.733	7.6

3.2.3. Correlations between attributes

Correlation refers to the relationship between two variables and how they may change together. It is very useful to know whether some pairs of attributes are correlated and how much. This help us to determine which variables should be included in the input to predict the output. The common various correlation coefficients are: Pearson correlation coefficient, Spearman's rank correlation coefficient and Kendall rank correlation coefficient. Pearson's correlation coefficient is the most common method to calculate the correlation in the literature. It measures the linear correlation between continuous variables. It has a value between +1 and -1, a correlation of 1 is perfect positive linear correlation, whereas a value of -1 is perfect negative linear correlation and 0 indicates no correlation.

For many machine learning algorithms correlated features might make poor performance. As such, it is important to review all of the pairwise correlations between the attributes in our dataset. Table 3 shows the correlation between all pairs of attributes calculated using Pearson's correlation method.

Table 3

Pearson's correlation

	CpuUsage	CpuUsagePer	MemCapacity	MemUsage	DiskWrite	NetTransmitted
CpuUsage	1	1	-0.229	-0.003	0.043	0.131
CpuUsagePer	1	1	-0.229	-0.003	0.043	0.131
MemCapacity	-0.229	-0.229	1	0.155	-0.011	0.01
MemUsage	-0.003	-0.003	0.155	1	0.026	0.043
DiskWrite	0.043	0.043	-0.011	0.026	1	0.263
NetTransmitted	0.131	0.131	0.01	0.043	0.263	1

In fact, this research is trying to select those features that have the strongest relationship with the output variable (CPU usage). From table (3) above, some correlations between CPU usage and memory usage and also with disk write can be seen. Furthermore, we need to exclude any redundant attributes before fitting the deep learning model, which save computation and time. Such as CPU usage and CPU usage percentage, they have perfect positive correlation with each other. In fact, the CPU usage percentage is calculated from CPU usage.

3.2.4. Visualization

It is an n excellent way to better understand the data is by generating plots and charts. Generally, data visualization helps to learn the data very fast and understand each attribute independently. Specifically, the following plots are produced:

- Histogram allows discovering the underlying distribution of a set of the data. Figure 6 shows the distribution of CPU usage of VM_1 which shows that CPU usage has almost Gaussian distribution, where most common observations are around the mean (137.29).

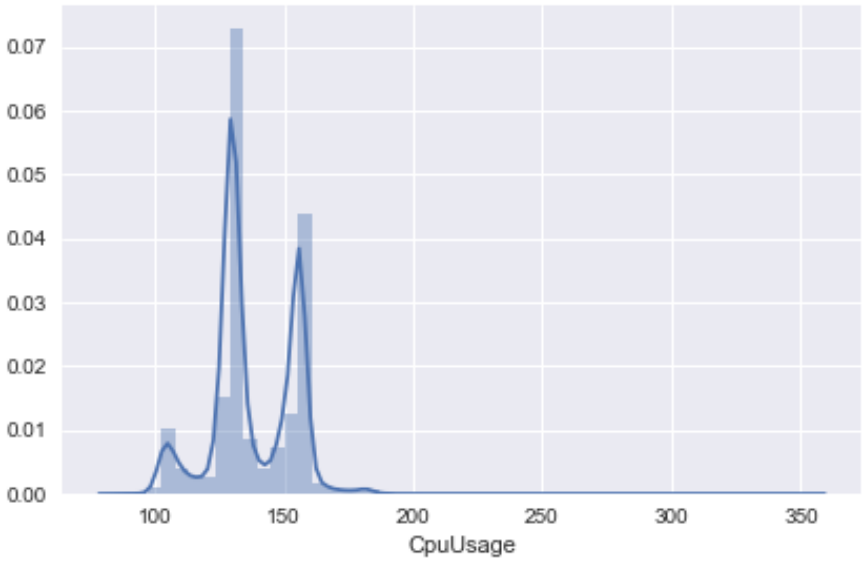

Figure 6. Histograms of CPU usage for VM1

- Correlation matrix which is used to explore the dependence between multiple variables at the same time. Figure 7 plot the correlation matrix for attributes in VM_1 which shows a positive correlation (0.13) between CPU usage and Network transmitted throughput.

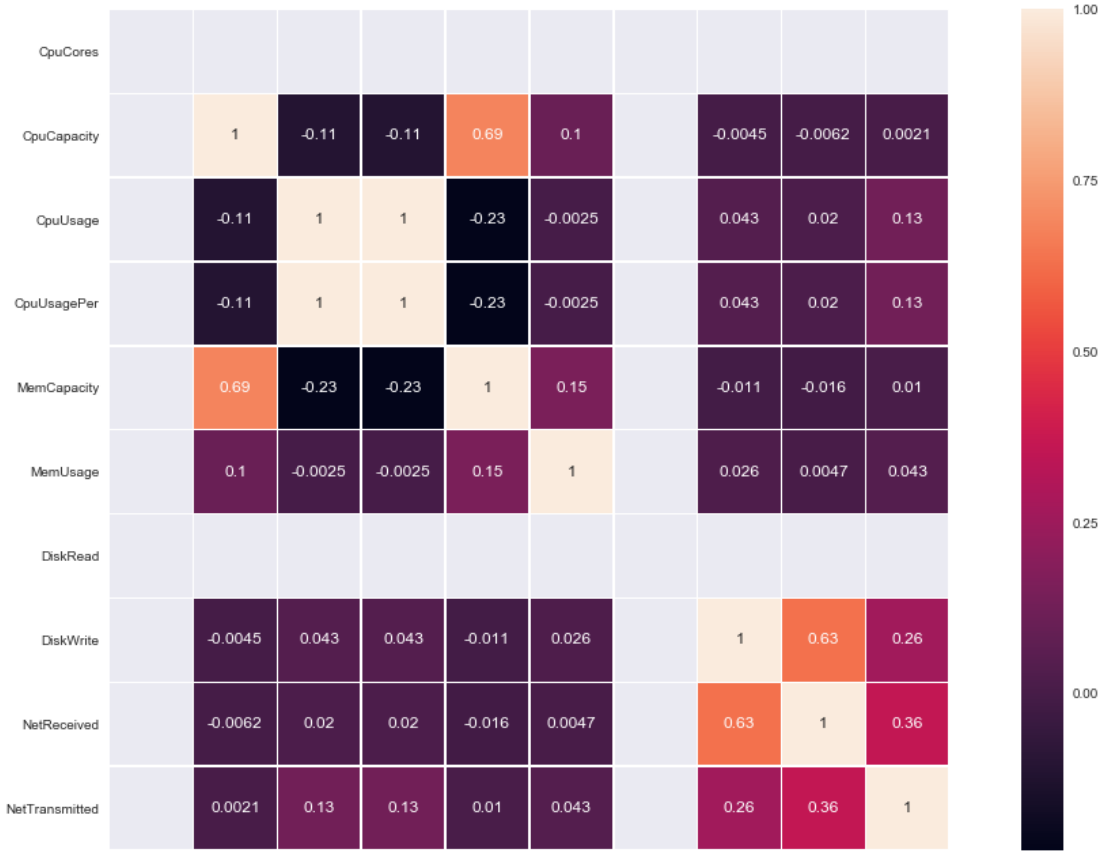

Figure 7. Correlation matrix for VM1 variables

- Pairwise joint plot that shows the relation of attribute pairs to examining their joint distributions. Figure 8 shows the joint plot between CPU usage and Network transmitted throughput features. The plot reveals a linear relationship between the two variables.

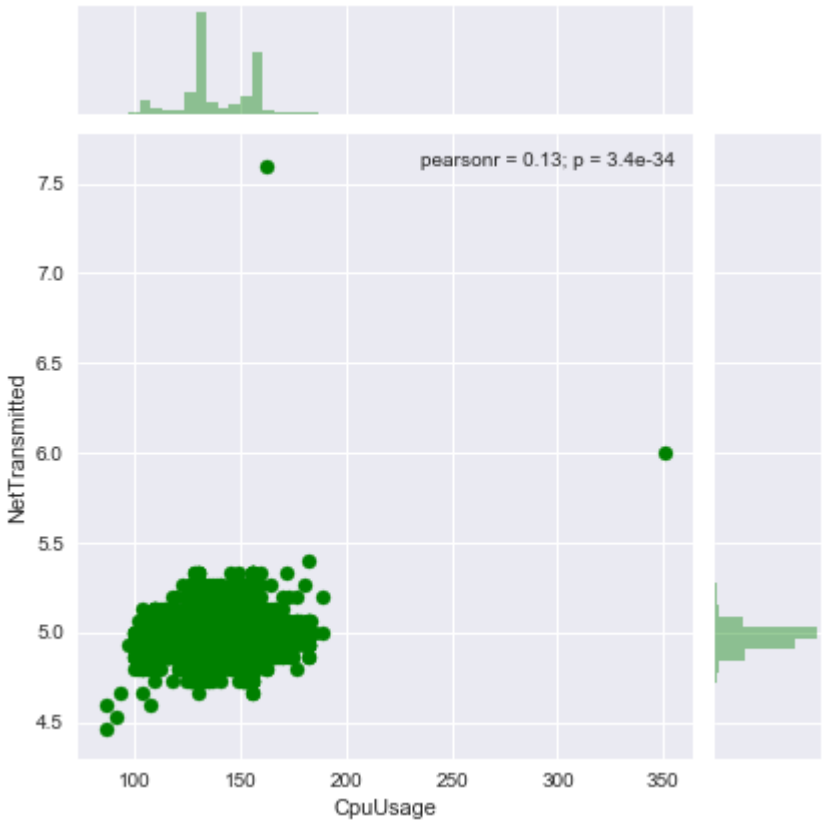

Figure 8. Pairwise joint plot of CPU usage and network transmitted features

3.3. Data Preprocessing

It is a very good practice to prepare the data in such a way so as to expose the structure of the problem to the deep learning algorithm that we are going to use. From exploring the data, we found that the dataset we have is clean and has no missing values. In this section we will describe the preprocessing phases we used to prepare the data for building the predictions models.

3.3.1. Features Selection

In this phase the redundant features might be removed and new features could be developed. In general, having unrelated features in the data can decrease the performance of many deep learning models. Feature selection helps reduce overfitting, improves the performance of the model and reduces the training time. Those features that contribute most to the output variable (CPU usage) need to be selected. For univariate deep learning model the CPU usage history will be used to predict the future values. For multivariate deep learning model it is important to select the desired features based on the data analysis done in the previous section. Memory capacity provisioned, Memory usage, Disk read throughput, Disk write throughput, Network received throughput and Network transmitted throughput will be used as input features. Figure 9 is a plot with 7 subplots showing 8621 observations from one month of CPU usage data. Other features will be removed as some of them are redundant (e.g. CPU usage in percentage) and others are fixed for all samples such as CPU capacity provisioned, CUP cores. Disk Read can be removed in this case as it has only zero values. The authors in [20] argue that their model works better than other approaches that rely on feature extraction and selective feature reduction, because they include all the features so that the model tries to understand the underlying patterns and dependencies.

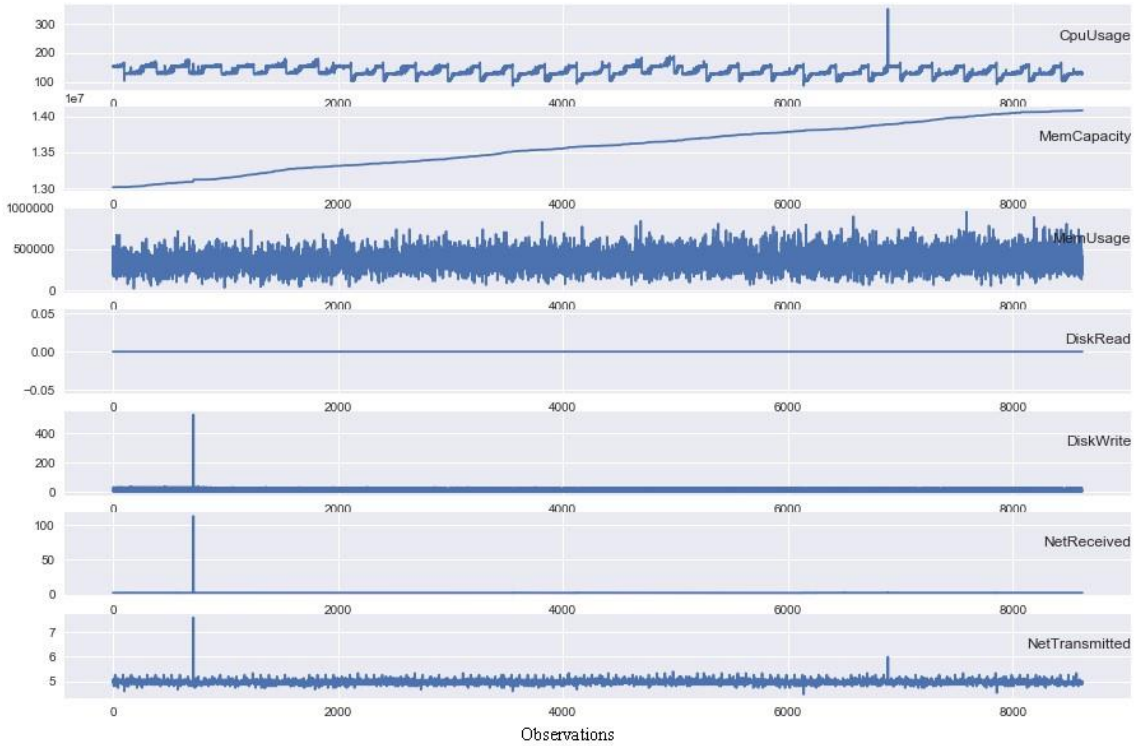

Figure 9. Line plots of VM1 time series

Feature selection can also be done automatically. In this case, the common method of recursive feature elimination (RFE) used to identify the features that are the most predictive. From simple conducted experiment, RFE helps to pick out the most important features as shown in table (4), where rank "1" means the most important to the output variable (the predictive importance decreases as the ranking increase).

RFE also shows that two features (Disk Read and Network received throughput) have the least importance. In fact, disk read for VM_1 can be eliminated. Truly, the results here conform with what we previously got in section 3.2.3.

Table 4

Ranked feature relevant for CPU usage prediction

Feature Name	Feature Ranking
Memory Capacity Provisioned	**1**
Memory Usage	**2**
Disk Read Throughput	**6**
Disk Write Throughput	**3**
Network Received Throughput	**5**
Network Transmitted Throughput	**4**

3.3.2. Stationary Data

Stationary time series means that the series of observations do not depend on time. In other words, stationary time series have no trend or seasonal components. Usually, when working with a sequence of real values, such as the data of this research making the series stationary has to be taken into account, if it is not stationary. Specifically, in such case the trend, seasonality or any other time-dependent structures has to be removed, if any is founded. Statistical modeling approaches assume the time series to be stationary. But, deep learning models (e.g. LSTM) have the ability to learn events for long time period. However, when the series is stationary it can be easier to model. In this case checking for stationary was made by two ways. First, by looking at the line plots of VMs workload. It was found here that there is an obvious lack of trend

and seasonality components for VM_1. Second, by use of statistical tests (i.e. reviewing summary statistics) like mean and variance to check if there is a change over time. The checking was performed by splitting the CPU usage series into two equal contiguous groups, then we calculated the mean and standard deviation for both groups. Table (5) shows the results for CPU usage variable in VM_1, which shows a little difference, but still in the same ballpark. As a result, we did not stationaries the data for deep learning model.

Table 5

Mean and variance of CPU usage variable in VM_1 for two equal partitions

Group 1		**Group 2**	
Mean	**Variance**	**Mean**	**Variance**
138.848465	231.151657	135.732864	269.071259

Generally, if there is a trend or seasonality, it can be removed by the standard differencing method, which helps to eliminate or reduce trend and seasonality [16]. Python Pandas library has a function that automatically calculate the 1st discrete difference dataset.

3.3.3. Data Normalization

Normalization referrers to process of rescaling the observations to have the same scale into the range between 0 and 1. Essentially, it is useful for sparse dataset that has attributes of varying scales. In our case it can be very useful as there are lots of zeros in

the dataset. Normalization is very useful for algorithms that weight inputs like neural network. Since this research is going to use the neural network Long Short-Term Memory recurrent neural networks all the features are normalized into the range [0 and 1] to be ready for univariate and multivariate models. MinMaxScaler class from scikit-learn [9]library was used to perform the normalization.

3.3.4. Lag and Sequences

Before using the data in any machine learning prediction problem, the data has to be reframed into supervised learning problem, which has the required format as input to the deep learning model. In this step the selected features from the previous step were transformed from their raw based time series format into lags and sequences. Lags represent the time steps needed to look back to the past, and each lag will be a feature that included for the deep learning prediction models. The sequences are the future time steps that are needed to predict. Initially, the next time step was predicted that represent the CPU usage for 5 minutes in the future. Since the prediction time interval is short the data was down-sampled to decrease the frequency of the samples from 5 minutes into 15 minutes and 45 minutes. Then for each of them the next time step was predicated, which represent 15-minutes and 45-minutes in the future. Table 6 below demonstrates a five lag time steps (t_1, t_2, t_3, t_4, t_5) to predict the current time step (t) for the feature CPU Usage. The input sequence in the figure is in the correct left-to-right order with the output variable to be predicted on the far right.

[9] http://scikit-learn.org/

Table 6

Reframing data to supervised learning problem (VM_1 data)

CpuUsage (t-5)	CpuUsage (t-4)	CpuUsage (t-3)	CpuUsage (t-2)	CpuUsage (t-1)	CpuUsage (t)
0.66372	0.658416	0.665842	0.665842	0.668317	0.663366
0.658416	0.665842	0.665842	0.668317	0.663366	0.665842
0.665842	0.665842	0.668317	0.663366	0.665842	0.673267
0.665842	0.668317	0.663366	0.665842	0.673267	0.668317
0.668317	0.663366	0.665842	0.673267	0.668317	0.673267

CHAPTER 4: METHODOLOGY AND DESIGN

This chapter describes the approach taken in this research project for predicting cloud resource needs. More specifically, this is a problem where given a historical resource usage data for a VM (e.g. CPU usage, Memory usage, Disk read/write throughput, Network received/transmitted throughput) the task is to predict the next CPU usage. This project, only predicts CPU usage. But, the same approach can be applied to predict other resource metrics.

The implementation process consists of several steps including: Loading Data, Preparing Data, Defining Models, Compiling Models, Training Models and Evaluating Models. Figure 10 shows an illustration of the implementation process. The process starts by loading VM historical data using python APIs, and carefully chooses the features that the research is interested in. Data is then preprocessed for training the models. First, all data are normalized into [0,1]. Then, the data is transformed from its row-based format into lag and sequence format. To evaluate the models on new unseen data, a common method for time series data is to split the dataset into train and test datasets. The dataset was split into training dataset and testing dataset with 75% of the observations used to train the models, leaving the remaining 25% for testing the models.

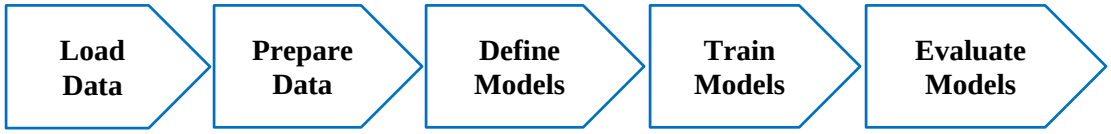

Figure 10. Methodology overview

Two approaches for VM recourse prediction based on deep learning and a statistical one are defined and created. The deep learning based approaches which are used in this project are MLP and LSTM, for each of them we build univariate and multivariate models. The statistical approach used in this research is the ARIMA model. The resource prediction models were initially defined, then trained on the training dataset. Finally, the models evaluated on the testing dataset to see how they perform on a new unseen data. Chapter 5 present the results of the experiments conducted to evaluate the mentioned resource prediction models.

Many experiments were conducted for each model to choose the best configuration parameters. Practically, selecting the number of layers to be used and their types are based on heuristics. Generally, there are some heuristics that we can use and usually the best network structure is found through trial and error process. Indeed, we need a network that is large enough to capture the structure of our prediction problem. Section 4.1 describes the design of MLP model. Section 4.2 describes the LSTM model. ARIMA model is described in section 4.3

4.1. MLP Model

We used Multilayer Perceptron using the window method so that multiple recent time steps get used to make the prediction for the next time step. Based on extensive experiments conducted we have chosen the following MLP deep learning configuration. For window size of the historical data we tried different values (2, 5, 10) and selected the size of window that gave best results which is five. The network topology is a fully-connected network structure with three layers chosen after trail of other deeper models (2,

3, 4). The first hidden layer has 50 neurons, the second hidden layer has 34 neurons, and finally the output layer has one neuron used to produce the output (the predicted CPU usage) many other wider models were tried. A summary of univariate and multivariate models and their configuration are depicted in Figures 11. For univariate MLP model multiple recent CPU usage feature was used to predict the next CPU usage. Whereas, for multivariate MLP model the problem was framed so that seven multiple recent performance metrics (see features selection on section 3.3.1) were used as input to predict the output CPU usage.

ReLU rectifier activation function was used on the first two layers as it is shown in the literature to give good performance [11]. No activation function was used for the output layer as it is a regression problem and the research is interested in predicting real values directly without transformation.

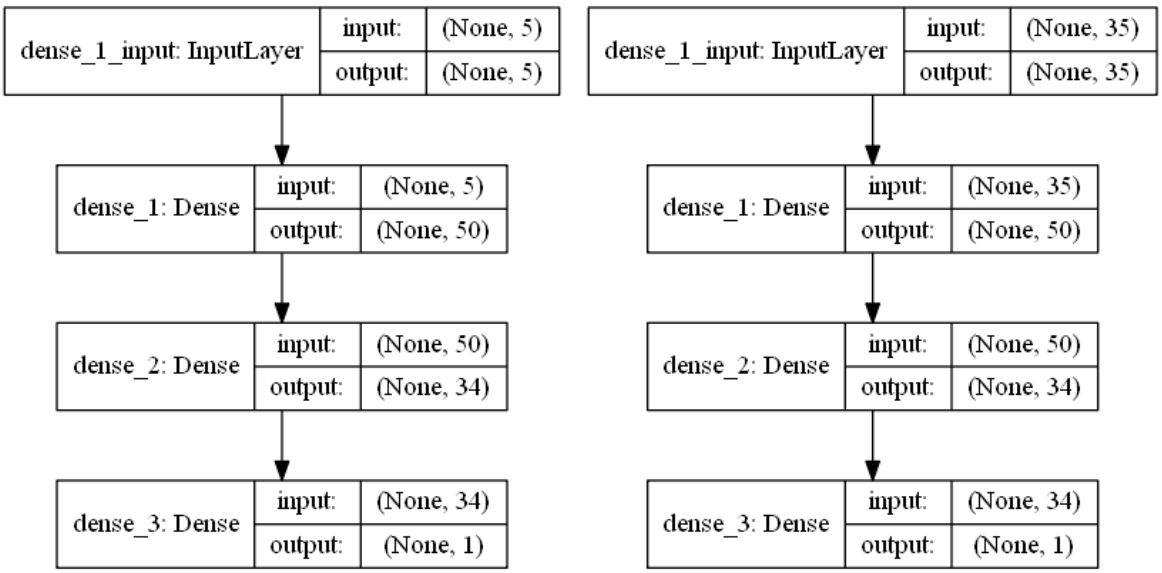

Figure 11. Univariate MLP (left), multivariate MLP (right) neural network models

After defining the univariate and multivariate MLP models. We need to compile them and specify the loss function and the optimizer. In general, compiling the model uses the numerical libraries under the hoods (i.e. back-end). In our case TensorFlow[10] was used. The loss function that used to evaluate a set of weight was Mean Squared Error (MSE). The efficient gradient descent algorithm Adam was the optimizer that had been used to learn the weights. Adam was selected, because it is an efficient default [31].

4.2. LSTM Model

The LSTM model architectures are shown in figure 12. The network architectures for both univariate and multivariate are made of two LSTM layers with two dropout layers of 20% after each LSTM layer. The dropout layer serves as regularization technique to prevent overfitting by randomly dropping out units with their connections. The last layer is the output layer. The first hidden LSTM layer has 100 memory blocks, or neurons and the second hidden LSTM layer has 50 memory blocks. The sigmod activation function is used for all LSTM memory blocks.

The Long-Short Term Memory using window method with time steps was used. LSTM model expects input to be provided with specific array format in terms of (samples, time steps, features), so that the preprocessed train and test data were transformed to be in the expected format (see section 3.3.4 for details). The window size of historical data is five (same as MLP models in order to have a far comparison later in the evaluation). For univariate models, five recent CPU usage used to predict the next CPU usage. But, for multivariate LSTM model, five recent steps for seven features (refer

[10] https://www.tensorflow.org/

to section 3.3.1 for more details about the selected features) were used as input to predict the output CPU usage. In general, LSTM models need to be configured in the right way, as they require a lot of preparation to get the data in the proper format for learning.

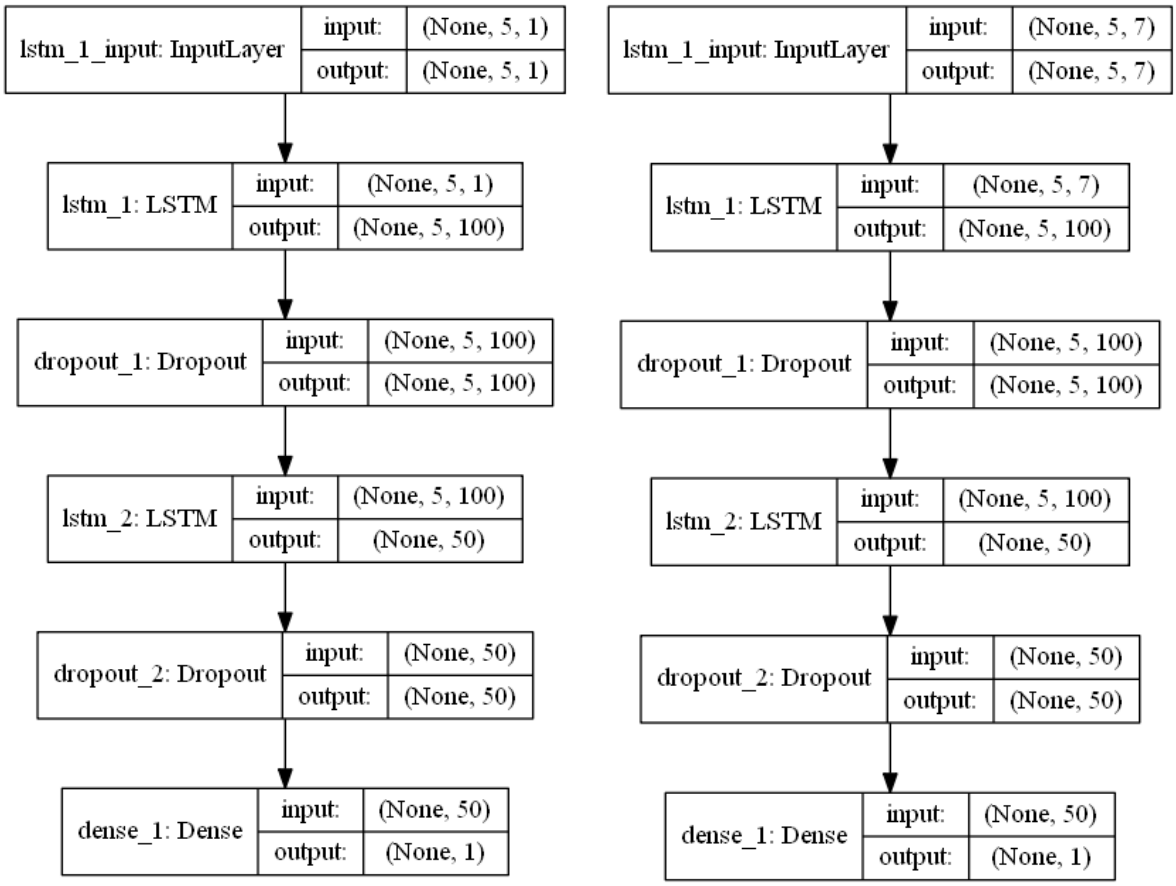

Figure 12. LSTM models architecture , univariate (left), multivariate (right)

For LSTM model compilation the same is used as MLP models. The loss function is mean squared error (MSE) and the optimizer is the efficient gradient descent algorithm Adam [31].

4.3. ARIMA Model

ARIMA is chosen because it is a common and broadly used statistical method for time series forecasting. The following parameters were used for ARIMA model (5,1,0), where the lag order for Auto Regression (p) was 5, the degree of differencing (d) was 1, and the order of moving average window (q) was 0. Figure 13 presents the ARIMA model coefficient values used to fit the training dataset.

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	959
Model:	ARIMA(5, 1, 0)	Log Likelihood	-3100.444
Method:	css-mle	S.D. of innovations	6.134
Date:	Sat, 28 Apr 2018	AIC	6214.887
Time:	04:59:48	BIC	6248.949
Sample:	1	HQIC	6227.859

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0262	0.221	-0.119	0.906	-0.460	0.407
ar.L1.D.y	0.4380	0.032	13.637	0.000	0.375	0.501
ar.L2.D.y	-0.1851	0.035	-5.279	0.000	-0.254	-0.116
ar.L3.D.y	-0.0221	0.036	-0.623	0.534	-0.092	0.048
ar.L4.D.y	-0.0291	0.035	-0.830	0.407	-0.098	0.040
ar.L5.D.y	-0.0980	0.032	-3.059	0.002	-0.161	-0.035

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0940	-0.8482j	1.3843	-0.1050
AR.2	1.0940	+0.8482j	1.3843	0.1050
AR.3	-0.2626	-1.6271j	1.6482	-0.2755
AR.4	-0.2626	+1.6271j	1.6482	0.2755
AR.5	-1.9593	-0.0000j	1.9593	-0.5000

Figure 13. Summary of the fit ARIMA model

CHAPTER 5: EXPERIMENTS AND RESULTS

This chapter describes the experiments conducted to evaluate three different prediction models namely Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA) using different workload data from a real distributed cloud datacenter that have different workload patterns namely periodic, growing and unpredicted workloads. Three common forecast error measurements were used to evaluate the developed resource prediction models. The experimental setup is presented in section 5.1. The results are shown in section 5.2 and discussed in section 5.3

5.1. Experimental setup

The experiments were performed on Intel® Core™ I7-M4712MQ processor of 2.30GHz clock speed, 4cores, 8 logical processors and having 16 GB of memory. Python along with Keras [11] library were selected as a tool for implementation and simulations. The experiments were performed on different 15 cloud VMs workload data from Bitbrain [12] for different workload patterns as mentioned in chapter 3 . It has been referred to them as VM_1, …, VM_{15}. These VMs represent different workload patterns. Authors in [17,18] classify workload patterns in cloud computing environment into three different types, namely periodic, growing and unpredicted workload. Each represents typical application or scenario. In this work the focus will be only on study CPU prediction for these workload patterns.

[11] https://keras.io/
[12] http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

The experiments were carried out on the workload data obtained from real data center to forecast resource needs. More specifically, we use the historical data to forecast the CPU usage for future 5, 15 and 45 minutes. As the evaluation of a model is an estimation on how well the model may do in practice we need to test the created prediction models on new unseen data and see how they perform. Hence, the data was divided in two parts called training and testing data in the ratio of 75:25. The 25% of the data were used to evaluate the model in which we make forecast then calculate the performance metrics with respect to expected values.

Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Square Error (RMSE) used to evaluate the created resource prediction models. Which are the most common evaluation metrics for time series prediction that focus on the prediction of real values [30].

For the deep learning based prediction models (MLP and LSTM), we developed two versions (univariate and multivariate). In addition to the classical statistical model (ARIMA), we can think of them as 5 prediction models, each is evaluated on 15 VMs that have different workload patterns (5 VMs for each workload type). The accuracy of prediction models in terms of MAE and RMSE are reported in the next section. The results are then compared and discussed to see which model is performing better for forecasting cloud resource needs.

5.2. Evaluation results

For comparing the results of the developed resource prediction models, the average RMSE (A-RMSE) and average MAE (A-MAE) are computed as the performance index. Table (7) and table (8) show the average RMSE and the average MAE per workload type of all prediction models for the CPU usage value for several future time intervals (5-minutes, 15 minutes and 45 minutes). The tables illustrate the workload type (WT), where "P" represents periodic workload, "U" represents unpredicted workload and "G" represents growing workload. The values in red color represent the best results for a given model for the forecasted interval.

Table 7

Average RMSE for prediction models (A-RMSE)

WT	LSTM						MLP						ARIMA		
	Univariate			Multivariate			Univariate			Multivariate					
	5	15	45	5	15	45	5	15	45	5	15	45	5	15	45
P	113.84	93.05	75.62	122.47	106.66	100.41	117.04	96.05	79.06	121.21	114.38	130.7	125.2	100.85	89.43
U	4222.95	5900.55	8617	5599.68	7611.59	15580.45	5014.45	6161.44	8951.17	6754.01	8532.01	13679.29	4106.02	5735.42	8544.12
G	635.82	652.64	792.79	791.49	1418.69	758.25	569.88	577.05	702.99	1516.44	1652.37	1310.31	465.09	484.42	535.43

Table 8

Average MAE for prediction models (A-MAE)

WT	LSTM						MLP						ARIMA		
	Univariate			Multivariate			Univariate			Multivariate					
	5	15	45	5	15	45	5	15	45	5	15	45	5	15	45
P	34.03	37.31	37.37	41.91	45.49	53.38	34.11	36.18	39.79	40.96	49.68	88.25	39.45	39.19	43.7
U	1702.39	2656.01	4341.94	2207.16	3049.49	6534.88	2315.58	2980.72	4500.79	3292.59	3634.99	6381.11	1323.46	2115.39	3697.77
G	360.05	310.51	546.61	463.63	575.02	462.27	348.95	285.86	434.78	537.06	645.28	697.61	204.38	207.55	241.82

Figures 14-18 show the accuracy of the prediction models of the predicted CPU usage compared with the actual (expected) CPU usage for the 45 minutes in the future for

VM_1. The x-axis represents the observations (every 45 minutes) while the y-axis represents the value of CPU usage. It can be seen that the predicted CPU usage using LSTM model are close to the actual CPU usage.

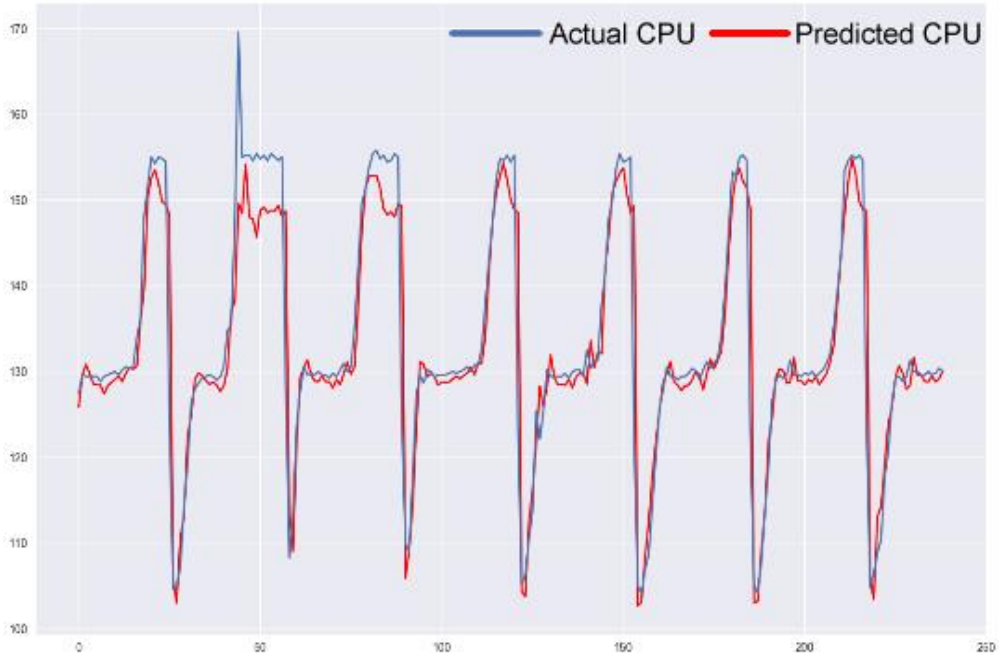

Figure 14. LSTM Univariate Model - Actual vs Predicted CPU usage

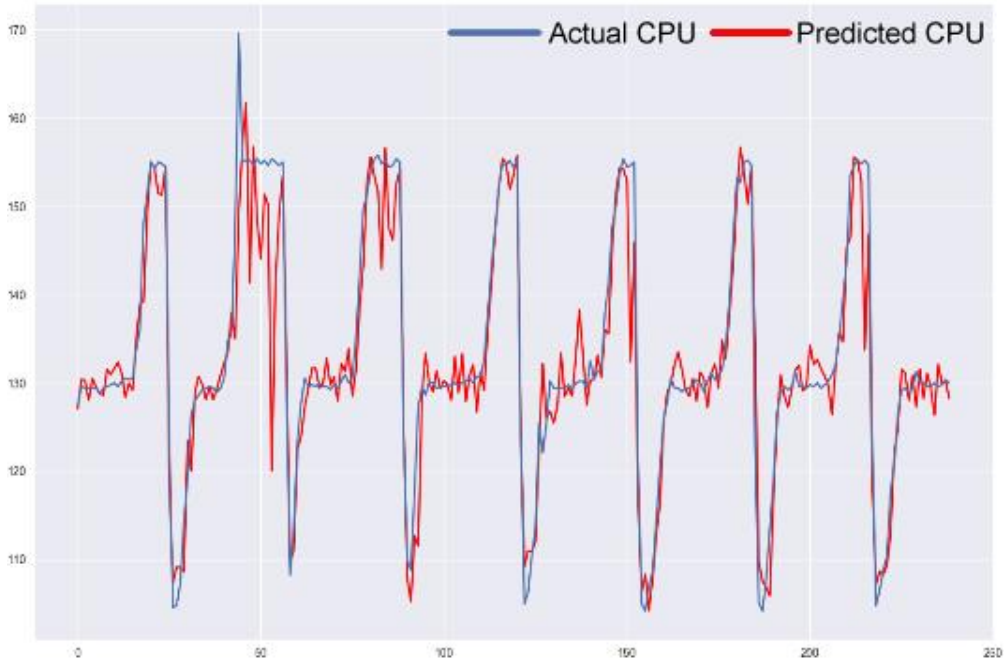

Figure 15. LSTM Multivariate Model - Actual vs Predicted CPU usage

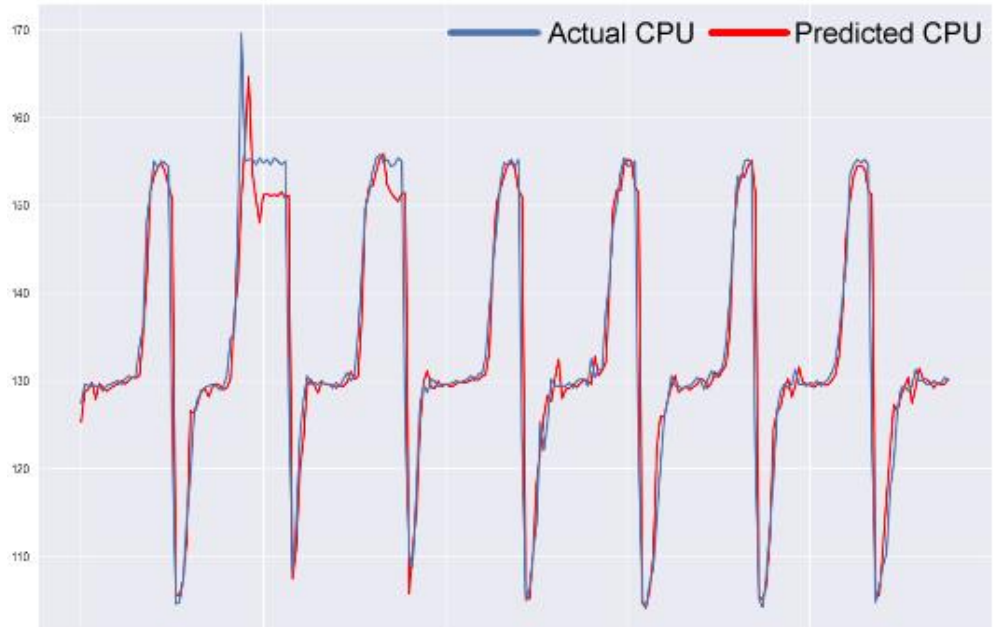

Figure 16. MLP Univariate Model - Actual vs Predicted CPU usage

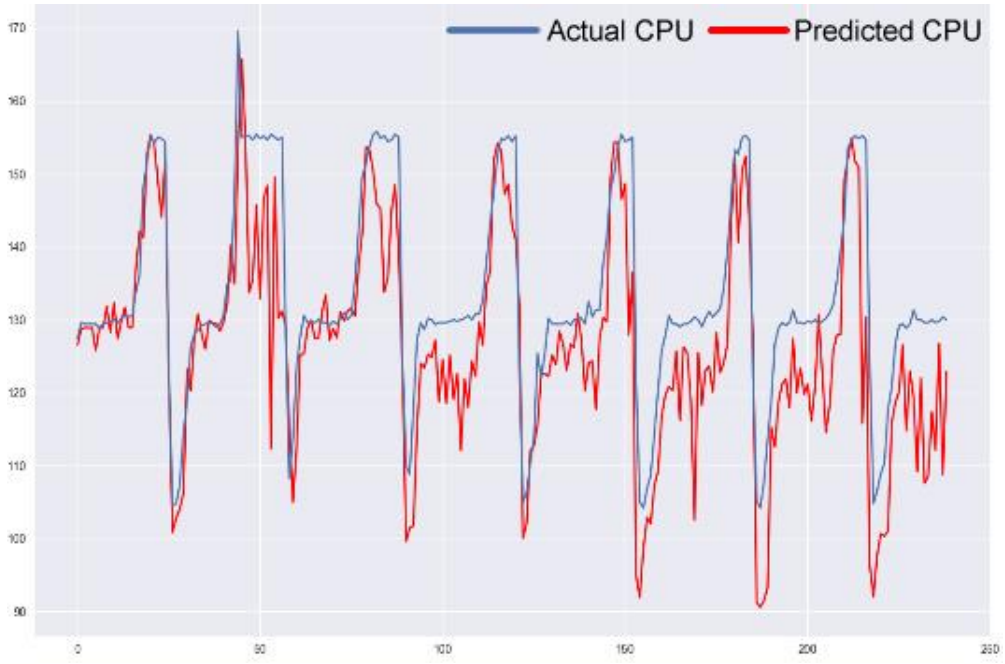

Figure 17. MLP Multivariate Model - Actual vs Predicted CPU usage

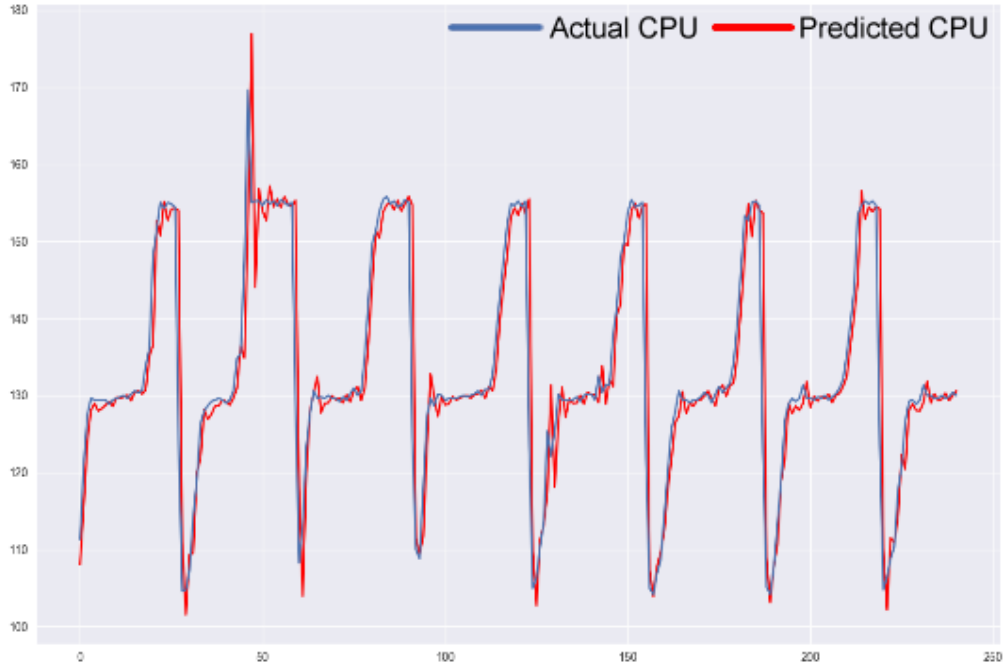

Figure 18. ARIMA Model - Actual vs Predicted CPU usage

5.3. Discussion

The outcomes of experiments have shown that the prediction accuracy of the developed resource prediction models (LSTM, MLP and ARIMA) depends on the incoming workload patterns. Evidently, there is no clear winner model for all workload patterns. However, the results clearly show that LSTM model outperforms other prediction models (MLP and ARIMA) for periodic workload patterns. It is evident from table (7) that LSTM model has better accuracy results, using RMSE for most of VMs that have periodic workloads (VM_1-VM_5). While the classical statistics model (ARIMA) has better accuracy results in growing and unpredicted workload patterns using both RMSE and MAE. Figure 19, presents the performance comparison of these prediction models for single VM CPU usage prediction.

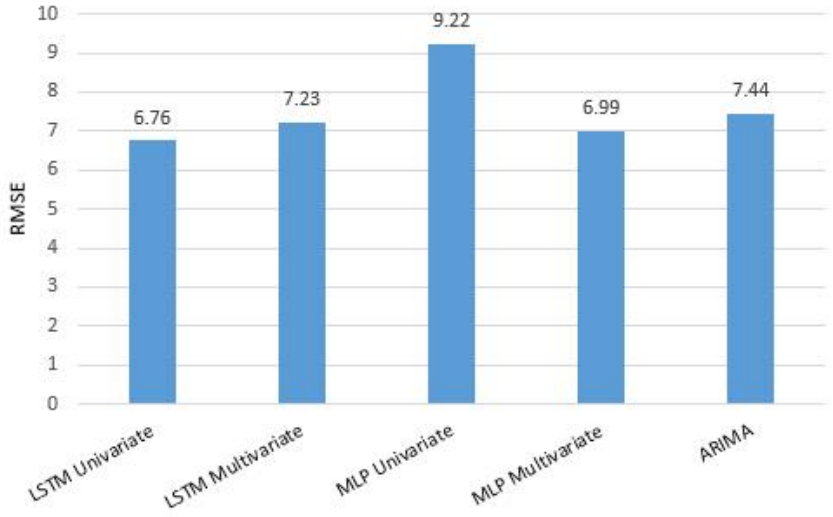

Figure 19. Comparison of VM_1 resource prediction (CPU usage)

Figure 20 and 21, illustrate the performance comparison of these prediction models for multiple VMs CPU usage prediction in 5-minutes time interval. Figure 20 shows the comparison for VMs that have periodic workload, it is apparent that LSTM univariate model has the best results with A-RMSE (113.84) and A-MAE (33.03), while figure 21 presents the comparison for VMs with growing workload, ARIMA proved to have the least A-RMSE (465.09) and A-MAE (204.38).

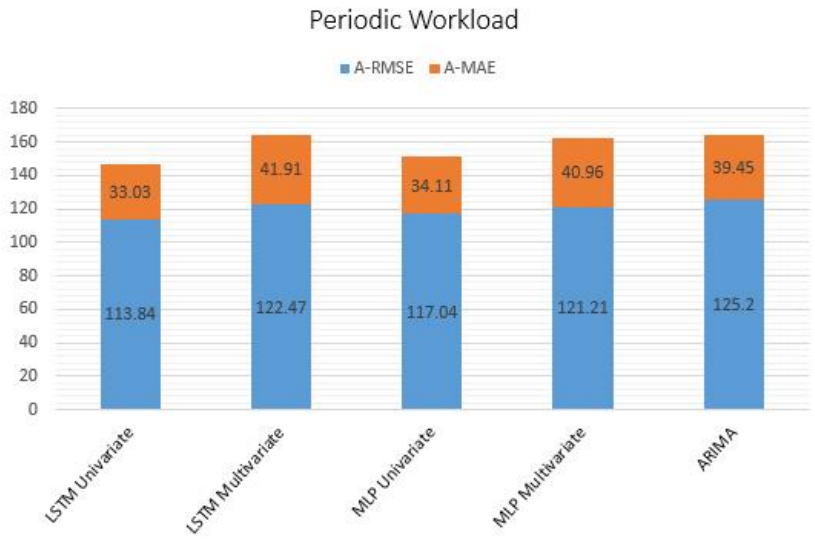

Figure 20. Comparison of multiple VMs resource prediction with periodic workload

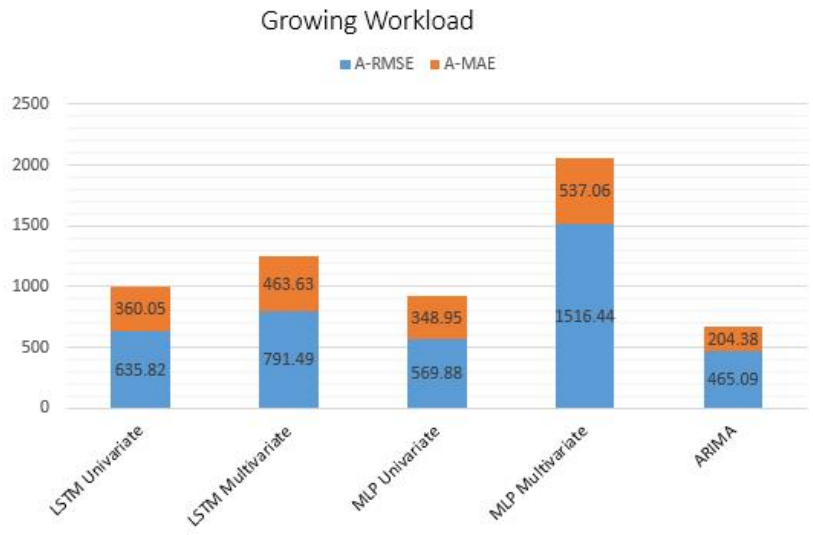

Figure 21. Comparison of multiple VMs resource prediction with growing workload

The results also have shown that the univariate models had produced more accurate results than multivariate models which mean there is no strong correlation between the used variables. In fact, this depends on the problem being solved.

Finally, based on the results of the experiments the following can be observed:

- LSTM is the best model for predicting future cloud resource needs for periodic workload pattern.
- ARIMA is the best model for predicting future cloud resource needs for the growing and unpredicted workload pattern.
- It is better to always start with univariate models and if they do not achieve good results then turn to the multivariate model, as in the univariate models single variable is observed, which saves computation and reduce the complexity of the model.

The eventual goal of this study is to improve prediction accuracy for proactive auto-scaling systems. Based on results, there is a great promise for LSTM to be used in analysis phase of proactive auto-scaling, as it produced the best results for periodic workload patterns despite of little tuning. LSTM produces better results with a large amount of data and enough training epochs, which is the case with cloud data. ARIMA model is also still a very good model, especially for a small dataset. In fact, ARIMA is one of the best state-of-the-art methods used in time series prediction. It is a combination of Auto Regression and Moving Average, also it adds the concept of integration to the model.

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this work two deep learning models (MLP and LSTM) and one statistical ARIMA model for predicting VM resource usage were designed, implemented and evaluated. The experimental results showed that the prediction accuracy of the developed resource prediction models depends on the incoming workload types and patterns. Precisely, the results showed that LSTM has better prediction accuracy for periodic workload patterns, while the statistical model (ARIMA) performs well on growing and unpredicted workload patterns. Based on these experimental results it is shown that LSTM deep learning can improve the accuracy of the CPU usage prediction for periodic workload compared to traditional prediction approaches. Moreover, it can be concluded that complex multivariate models do not always perform better. Our results show that univariate models achieved more accurate predictions.

In future work, we can explore algorithms that automatically characterize the VM workload to enable selecting the most suitable resource prediction model based on the workload patterns. Moreover, the features selection can be done automatically and this enables to include the most important features as input to the prediction model in order to obtain more accurate results.

REFERENCES

[1] Jennings, B., & Stadler, R. (2015). Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 23(3), 567-619.

[2] Chaisiri, Sivadon, Bu-Sung Lee, and Dusit Niyato. "Optimization of resource provisioning cost in cloud computing." IEEE Transactions on Services Computing 5.2 (2012): 164-177.

[3] Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559-592.

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

[5] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26.

[6] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA.

[7] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1(1), 7-18.

[8] Deng, L. (2012). Three classes of deep learning architectures and their applications: a tutorial survey. APSIPA transactions on signal and information processing.

[9] Shen, S., van Beek, V., & Iosup, A. (2015, May). Statistical characterization of business-critical workloads hosted in cloud datacenters. In Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on (pp. 465-474). IEEE.

[10] The Grid Workloads Archive (http://gwa.ewi.tudelft.nl)

[11] Brownlee, J. (2016). Deep learning with Python. Gumroad, Melbourne.

[12] Amiri, M., & Mohammad-Khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. Journal of Network and Computer Applications, 82, 93-113.

[13] Brownlee, J. (2017). Long Short-Term Memory Networks with Python.

[14] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2014, July). Measuring prediction sensitivity of a cloud auto-scaling system. In Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International (pp. 690-695). IEEE.

[15] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2015, May). Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (pp. 35-45). IEEE Press.

[16] Hyndman, R. J., & Athanasopoulos, G. (2014). Forecasting: principles and practice. OTexts.

[17] Calzarossa, M. C., Della Vedova, M. L., Massari, L., Petcu, D., Tabash, M. I., & Tessera, D. (2016). Workloads in the Clouds. In Principles of Performance and Reliability Modeling and Evaluation (pp. 525-550). Springer, Cham.

[18] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2015, May). Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (pp. 35-45). IEEE Press.

[19] Hernández, E., Sanchez-Anguix, V., Julian, V., Palanca, J., & Duque, N. (2016, April). Rainfall prediction: A deep learning approach. In International Conference on Hybrid Artificial Intelligence Systems (pp. 151-162). Springer, Cham.

[20] Gope, S., Sarkar, S., Mitra, P., & Ghosh, S. (2016, July). Early prediction of extreme rainfall events: a deep learning approach. In Industrial Conference on Data Mining (pp. 154-167). Springer, Cham.

[21] Huang, W., Hong, H., Li, M., Hu, W., Song, G., & Xie, K. (2013, December). Deep architecture for traffic flow prediction. In International Conference on Advanced Data Mining and Applications (pp. 165-176). Springer, Berlin, Heidelberg.

[22] Jheng, J. J., Tseng, F. H., Chao, H. C., & Chou, L. D. (2014, February). A novel VM workload prediction using Grey Forecasting model in cloud data center. In Information Networking (ICOIN), 2014 International Conference on (pp. 40-45). IEEE.

[23] Jiang, Y., Perng, C. S., Li, T., & Chang, R. N. (2013). Cloud analytics for capacity planning and instant vm provisioning. IEEE Transactions on Network and Service

Management, 10(3), 312-325.

[24] Bankole, A. A., & Ajila, S. A. (2013, May). Predicting cloud resource provisioning using machine learning techniques. In Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on (pp. 1-4). IEEE.

[25] Calheiros, R. N., Masoumi, E., Ranjan, R., & Buyya, R. (2015). Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Transactions on Cloud Computing, 3(4), 449-458.

[26] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3–4), 197-387.

[27] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2017). An autonomic prediction suite for cloud resource provisioning. Journal of Cloud Computing, 6(1), 3.

[28] Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01 (pp. 193-200). Springer, London.

[29] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[30] Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., & Kamaev, V. A. E. (2013). A survey of forecast error measures. World Applied Sciences Journal, 24, 171-176.

[31] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.