Resource usage prediction on a cloud using Deep Neural Network

07 June 21

Student's Name: Jaydeep Gami
Student ID: 19868705
Course: Master by Coursework in Computer Science
‡ Department of Computing, Curtin University, Western Australia
Email: j.gami@postgrad.curtin.edu.au

* * *

Supervisor: Sonny Pham
Co-Supervisor: N/A

June 8, 2021

Abstract

Infrastructure-As-A-Service (IAAS) is one of the emerging services of a Cloud Computing that provides virtual computing resources like storage device, network connection, hardware to the user on-demand in an elastic manner. But many challenges occur while using on-demand elastic services. The main challenge is a delay in resource initialization, when Any user tries to access the IAAS services it takes some time to initialize the service on a cloud. The delay of time very important while running business-related workloads on a cloud i.e web servers, storage server, mailing service. Prediction of the business-related workload provides better system utilization and optimal computing result. However, the challenging task to predict the workloads on a cloud since the loading of resources dramatically fluctuate in a short period. In Proposed model use Long-Short Term Memory(LSTM) to predict the CPU utilization on a cloud, it uses the GWA-T-12-Bitbrain [1] dataset provides 1750 VM performance metrics. Comparative analysis of the proposed model with a time-series model provides a greater view of the performance of resource prediction.

Keywords: Cloud Computing, LSTM, Time-Series, ARIMA, Neural Network, resource prediction

Content

Chapter 1

Introduction

Cloud computing is one of the prominent computing technology that provides many services in a distributed environment. Based on the inclusion of extensive contextual foundation, Cloud Computing is classified into 3 services, Infrastructure-As-A-Service (IAAS), Platform-As-A-Service (PAAS), Software-As-A-Service (SAAS) [2]. The prominent feature of cloud computing is Virtualization. One of the big advantages of cloud computing is to provide all IT resources related to network, hardware, and storage on demand without any physical setup. Customize the system based on need and demand easily attach or detach the resource services. However, one of the flaws of this technology that when demand increase or too many requests to access the resources on a cloud then it takes some time to initialize the new resources which affects the performance and accessibility of the system. The purpose of predicting the resource usage in cloud computing to utilize the resources while managing workload between resources to minimize the execution time and improve the efficiency of the system on a cloud.

Over-provision of resources one of the main challenges in cloud computing. When any system is deployed in a cloud environment, it reserved some resources to manage the future workload. Reservation of resources helps to end-user to access the system during peak hours and improve the performance of the system but during the non-peak time, it is a wastage of resource which cost more to business vendors. If the resources are under-provisioned then there might be a chance of unavailability of services due to a spike in user requests. Lack of strategy and poor planning of scheduling resources during peak time may produce inefficiency in the performance of the system that directly affects the business that is running on a cloud environment [3].

Achieving the high performance of a system running on a cloud is very complex. Proper resource allocation not only improves the efficiency of the system but also minimizes the cost of resource usage on a cloud. Therefore, any proposed model improving the efficiency of performance without any economic benefits not suitable in the environment.

Optimal allocation of resources is one of the good strategies to improve performance. Scheduling of the resource is the key factor to minimize cost. There are important key factors consider while scheduling the resources i.e., demand of a resource, availability of existing resources, user requests workload on a system, scheduling algorithm. The main of scheduling to provide service to end-users with minimum response time with minimal cost. The proposed solution improves the efficiency of the performance which minimizes the cost of resource usage on a cloud [4].

Armbrust et. al mentioned in their paper the inflexibility of cloud computing is to predict the resource utilization of systems [5]. One of the difficult tasks is to perform a resource prediction algorithm on the fluctuation of data, most of the computing tasks of any system perform on the virtual machine on a cloud. There are many key metrics i.e., CPU usage, memory usage, network throughput, CPU cores, network capacity, generated by a virtual machine that helps to predict the demand of workload and based on-demand scaling the resources. There is a traditional time-series model like Auto-Regressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA) helps to forecast the resources based on demand[6]. But the traditional timer-series approach may not provide good results in multi-variant data systems or when there is the most significant amount of variation in historical data.

Neural Network is the key concept that solves the above problem, from past decades, Machine learning is very useful to solve the prediction problem in cloud computing[5, 7]. There are many traditional machine learning algorithm provides measurable result but the most effective and accurate method is Neural Network (NN). The Neural Network model is the mimics of the brain neuron cells, based on several neurons passing information to the next layers that perform the complex calculations for prediction. CPU utilization is one of the important factors while a system running in the virtual machine on a cloud. The proposed model uses a Long-Short Term Memory (LSTM) recurrent neural network to predict the CPU usage, and comparative analysis of results with previous time-series models.

Chapter 2

Literature Review

Many resources consider as an important factor while predicting cloud workloads CPU usage, Memory Usage, Network throughput, etc. it is a challenging task to predict the constant fluctuation in the resources for a short period (e.g., 10 to 30 min). Two approaches help to predict resource usage on a cloud. The first is a time-series analysis which helps to monitor continuous data and based on previous data forecast the future resource usage. Another approach is to machine learning and deep learning model that use a complex calculation based on previous data and predict the resource usage. In this paper, first, we are performing time-series analysis on the BitBrain dataset and after that apply LSTM Neural Network to predict the future resource usage [7].

2.1 Time-Series Analysis

In Virtual machines, the CPU is one of the common metrics that studied while analyzing the performance of the system. Dinda et. al. mentioned their article compare different time-series models (i.e., ARIMA, AR, MA, ARIMA, ARMA) based on CPU load. While comparing every time-series model they found that Auto Regression (AR) model with 16 or higher number of order efficiently forecast the 1hz of data up to 30 seconds in the future. The performance of the model is better compared to other model and provide optimal fit to data. AR model providing a very small amount of CPU and network throughput on 1hz of data and produce up-to-date forecasting for the system [8].

Parminder Singh et. al. introduced the TASM model which is a technocrat ARIMA and Support Vector Regression (SVR) machine used to predict workload on web applications available on the cloud. To check the performance of the system they used two weblogs on the proposed model and perform a comparative analysis of the other time series model (i.e Naive forecast Model, MA, ARIMA, ARMA, AR, SVR). The proposed model provides efficient result in both seasonal pattern and non-seasonal pattern data.

While comparing all the error metrics that are applied on TASM and other time series model, the author found that TASM provide efficient results in all error metrics. When TASM applying on one of the weblogs that produce 47.76 MAE which is optimal compare to another model (i.e.Naive:70.02, ARIMA:80.36, ARIMA:66.39, SVR:66.77 MAE). TASM model significantly provides optimal result and improve the performance of the web application available on a cloud, as a results model mitigate cost and energy on a cloud [9].Vazquez et. al. mentioned in their paper how to apply different time series models to forecast workload on a cloud data center for resource provision. In this paper, the authors use several time series models (i.e., First-order AR, First Order MA, ARIMA, Neural Network Auto Regression Model) on Intel Netbatch logs and google cloud trace. while applying models on both datasets, they found that for a small period like 10 min forecasting models are less predictable compare to a 1-hr period time. While performing forecasting model on Intel Netbatch, AR method produces very low error 27.70% error while moving average model produces the highest error 40.85% MAPE during a 10-min period. While ARIMA and NNET model produce 29.72 and 29.75 MAPE respectively[10].

Calheiros et. al. [11] proposed the ARIMA model to solve the workload prediction for cloud-based Software-As-a-Service (SAAS) and how to impact the quality of service of the application. The authors mentioned that dynamic resource provisioning affects the Quality of Service (QoS). if users access the service during peak time and because of static resource provisioning user may get delay while accessing the service in this particular situation Quality Of service is poor which affects the user experience on service. ARIMA model predicts the future workload based on the real-time traces of requests to web application server. ARIMA model able to reach up to 91% accuracy, as a result, produce greater efficiency in resource usage and Quality of Service.

2.2 Neural Network model

For the last decades, machine learning provides extensive results in the field of future Prediction based on historical observation and complex calculations. To provide a more accurate result Neural Network is the key that computes a complex calculator on each node of the layer and passes the information to the next layers of nodes. It creates a complex network; each succession node receives information from all previous nodes and computes calculators like mimics of brain cells.

Zhu et. al. [12] proposed a model of an Attention-based LSTM encoder-decoder network that helps to predict future workload on a cloud. The proposed model is divided into two parts, one with the LSTM encoder-decoder and another one is output layers. First

input data pass to the encoder of LSTM converts all input data into context vector by extraction of contextual and sequential feature from the input data. Then decoder of the network decodes the context vector and predicts the future workload in the batch. Output layers convert information coming from the Decode layer to the final prediction value of the model. The proposed model used 64 dimensions hidden state in both encoder and decoder, after prediction, apply on Alibaba cluster trace 2018 and perform comparative analysis to check the efficiency of the model. ARIMA model generates 33.126% MAPE error, Gated Recurrent Unit Encoder-Decoder (GRUED) model produce 24.362% MAPE while our proposed model has 19.529% MAPE. the proposed model produces state-of-art performance also scroll prediction method allows reducing the error which generates during long-term prediction and divide tasks into smaller tasks.

Gupta et. al. mentioned in their paper how Bi-Directional LSTM helps to improve the performance in workload prediction in cloud computing [13]. The traditional time series model only able to save information about the latest past data. In Neural Network when found the local minima, it's looking for global minima by iterating the local minima which take a small step towards the negative function of gradient, which makes the learning process slow this is known as the vanishing gradient problem. To learn long-term dependency between the data LSTM is an efficient model that provides memory cells that help to store the dependency and is widely used in non-linear problems. However, LSTM only uses one direction to process the data which leads to an overfitting problem in some datasets. The proposed model is Bi-LSTM that uses 2 LSTM to work Parallel. Bi-LSTM eliminates the problem of overfitting using two hidden data layers that work in both directions. In the paper, the Author produce prediction and an analysis result with 10min, 20min, 30min interval steps, And perform comparative analysis to check performance with another model (i.e. ARIMA, LSTM -U (univariate), LSTM-M(multi-variate), Bi-LSTM-U (univariate)) with the proposed model, for 10 min of interval BILSTM with multivariate produces 0.0095 RSME of CPU usage prediction while other models like ARIMA produce 0.0198, LSTM-U with 0.00123, LSTM-M generates 0.00115, Bi-LSTM-U produces 0.00105.while for 20-min and 30-min both produce effective results of the 0.00184 and 0.00225 respectively. In the end, the author concludes that Bi-LSTM produces state-of-art performance compare to LSTM and Bi-LSTM-U prediction models.

Ouhame et. al. [14] produced CNN-LSTM (Convolution Neural Network with LSTM) architecture to predict the resource utilization of cloud data centers. They also performed the comparative analysis of error metrics with a different hybrid model like VAR-MLP (Vector Auto Regression with Multi-Layer Perceptron), VAR-GRU (VAR with Gated Recurrent Unit), ARIMA-LSTM. Based on comparison they found that VAR-MLP produced 0.3446, VAR-GRU generates 0.3295, ARIMA-LSTM with 0.3111,

the and proposed model produced 0.3193 RMSE error on test data. In proposed model have 80% training data and 20Shen and Hong [15] used a Bi-directional model to predict host load in cloud computing. They used one month trace of the google data center [16] and based on CPU usage of all tasks running on the VM machine predict the future workload on the VM. Also compare performance metrics with the different models like LSTM ED(Encoder-Decoder), LSTM, AR (Auto Regression), ANN (Artificial Neural Network). The proposed Bi-LSTM used 24/64 size input layers,128 units for hidden layers, batch size with 128, global clipping norm rate with 5, 90 epochs, 10 early stopping rates, and 0.0.1 dropout rates. Janaradhan and Barrett [17] mentioned about CPU workload prediction on cloud data centers using ARIMA model and LSTM model. ARIMA model performed on google data trace and based on value of (2,1,2) of (p,d,q) it generates error between 37.331% to 42.881% on different VM machines. For LSTM model, if the data is not stationary then applied first difference to stabilize the data without loose of information. Then tanh activation applied to normalized the data from -1 to 1. The output of RMSE error of LSTM model is 0.0317 on training data and 0.0381 on testing data. After multiple change in hyperparameter of LSTM model authors found that LSTM model predict the workload with 17.566 to 23.65% error rate. Hence for trace of google data Authors conclude that LSTM is excellent model for predict the future CPU usage.

Table 2.1: Summary of Literature Review

Prediction Techniques	Dataset	Error Prediction	Preprocessing data	References
Attention based LSTM encoder-Decoder	Alibaba and Dinda workload traces	MAE,RMSE, MAPE	Yes	[12]
Bi-Directional LSTM	Google data traces	RMSE	Yes	[13]
CNN-LSTM	BitBrain dataset	RMSE,MAE,MSE	Yes	[15]
Bi-LSTM	Google data trace	MSSE,MSE	Yes	[16]
LSTM and ARIMA	Google data trace	RMSE, MAPE	Yes	[17]

Chapter 3

BACKGROUND

Time-series data refer as observation that collects over different period of time, Due to collection of data sampled during period of time there is chance of correlation between those observations. Time-series forecasting use past data to predict the future values. Time-series graph helps to identify the following characteristics

1. Seasonality
2. Trends
3. levels
4. Residual

In time series data seasonality is a pattern of observations that occurs after particular time on repeated manner. it could be weekly, monthly, yearly or hourly bases. Trends existing when there is a constant linear upward or downward behaviour in the observation over period of time. For time-series analysis data should be stationary.

What is Stationary in Time-Series model?

In time-series stationary is the process in which mean, autocorrelation and standard variation of observation constant over the period of time. There is assumption about time-series that a time-series data is stationary. Stationary time-series data produce meaningful prediction of data. Consider the example of rainfall data which is different over the year, but the average of rainfall equal to small amount of year almost identical which produce the plausible result. There might be chances that for long period of time average of rainfall is not feasible i.e. increase in climate change. Indeed, statistical analysis of time-series under the typical assumption of stationary of observations. In real world some time-series dataset is not stationary hence adjustment or elimination of the trends and seasonality must be required. The common approach to deal with non-stationary data to apply the difference of time-series data /citerao2008course.

3.1 Classic Time Series Model

3.1.1 ARIMA model

In the time-series analysis white noise is the process where variation and mean of the observation are not constant or set to zero, hence the correlation between its observation is different. Since data is uncorrelated, previous observation is not useful to predict future data. Moving Average and Auto Regression is the model that helps to eliminate these violations in time series. ARIMA is the combination of Auto Regression (AR) with Moving Average (MA) with Integrated (I) module.

- AR: Autoregressive is a model that current value considers as Xt calculated with a combination of past information n values, Xt-1, Xt-2 Xt-n with white noise (random error) Wt.

  $Xt = \Phi_1 Xt - 1 + \Phi_2 Xt - 2 + \Phi_3 Xt - 3 \Phi_n Xt - n + Wt$

- MA: AR model only uses past information of n values like Xt-1, Xt-2, ... Xt-n to forecast the future value with the current value of error Wt. AR model not used correlated error value of the past observations. Moving Average model is predicting the future data based on the previous value of error. Consider if you want to predict the Xt calculated by a combination of past 'n' values of errors, Wt, Wt-1, Wt-2, ... Wt-n.

  $Xt = \Phi_1 Wt - 1 + \Phi_2 Wt - 2 + \Phi_3 Wt - 3 \ldots\ldots \Phi_n Wt - n + Wt$

- I: Integrated is a process when is useful when data is not stationary. If data is not stationary then the difference of each observation is calculated with the previous observation of timestamp and produces stationary time series data.

Each of these parameters set in the model, a standardized notation represent as ARIMA (p,d,q) where each parameter substitute with different integer values.

- p: p denoted as lag order in Autoregressive that helps to select several past values to refer for predicting future data.
- d: d is denoted as a difference that helps to use when data is not stationary. If data is not stationary after applying difference, then increase the degree of difference to stationary the time series data.
- q: q is denoted as a window size which normally provides a total number of lagged error of window size for forecasting.

$$
\begin{aligned}
x_{1t} &= \phi_{11}x_{1,t-1} + \phi_{12}x_{2,t-1} + \epsilon_{1t} \\
x_{2t} &= \phi_{21}x_{2,t-1} + \phi_{22}x_{2,t-2} + \epsilon_{2t}
\end{aligned}
$$

where $E(\epsilon_{1t}\epsilon_{2s}) = \sigma_{12}$ for $t = s$ and zero for $t \neq s$. We could rewrite it as

$$
\begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ 0 & \phi_{21} \end{bmatrix} \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \phi_{22} \end{bmatrix} \begin{bmatrix} x_{1,t-2} \\ x_{2,t-2} \end{bmatrix} + \begin{bmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{bmatrix},
$$

or just

$$
\mathbf{x}_t = \mathbf{\Phi}_1\mathbf{x}_{t-1} + \mathbf{\Phi}_2\mathbf{x}_{t-2} + \boldsymbol{\epsilon}_t
$$

and $E(\boldsymbol{\epsilon}_t) = \mathbf{0}$,$E(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_s) = \mathbf{0}$ for $s \neq t$ and

$$
E(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t') = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}.
$$

FIGURE 3.1: Example of VAR model

3.1.2 Vector Auto Regression (VAR)

Vector Auto Regression is one of the prominent models in multivariate time-series data in which extension of the single univariate AR model to multivariate time-series AR model. In the VAR model, each observation of the time series data not only depends on own its lag but also depends on the past observation of the data. A simple expression of the VAR model is given below

$$Xt = \Phi + \Phi_1 Xt - 1 + \Phi_2 Xt - 2 + \Phi_3 Xt - 3\Phi_n Xt - n + Wt$$

3.2 Neural Network model

Artificial Neural Network (ANN) is the mimic of a design of the human brain system. Similar like brain neurons an ANN use different layers of neurons which are connected perform complex computation task. The main difference between a single feed-forward network and RNN is there is one layer which is known as the recurrent layer that helps to store information about past information. Figure 3.2 describes the recurrent network cell that helps to store long-term dependency of the input sequence in their recurrent layers. RNN is not useful while handling long-term dependency. Vanishing gradient or exploding the gradient such kind of phenomena encountered due to long-term dependency.

RNN performs backpropagation through time to adjust the weight and value of the hidden layers to minimize the loss function between true labels and predicted values. It improves the model using gradient descent which uses a learning rate to improve the model, if a learning rate is too low then multiplication and the complex calculation that

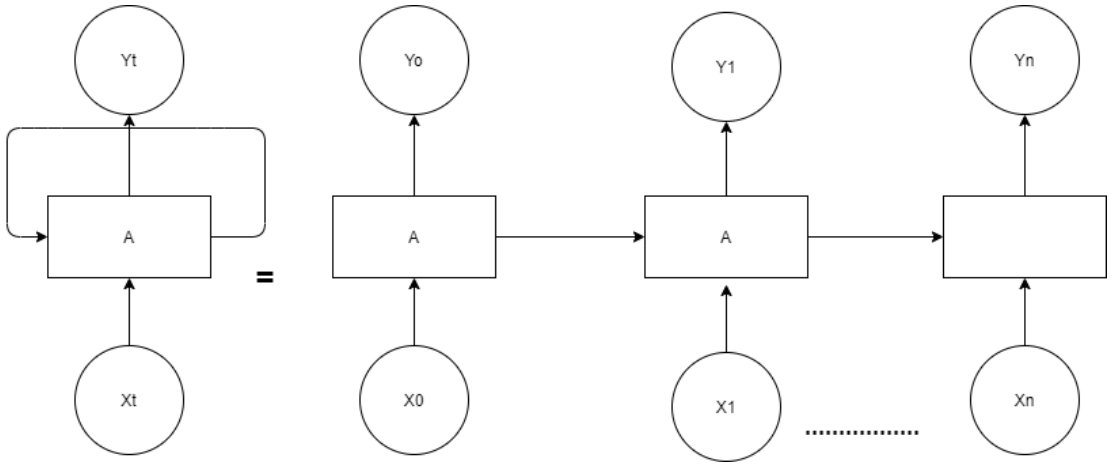


FIGURE 3.2: Recurrent Neural cell

is performed during each layer, hence vanishing gradient situation occurred after a long period. If the learning rate is too high then it misses global minima and exploding gradient after a long period. LSTM model solves the vanishing gradient problem [18].

3.2.1 Long-Short Term Memory Model

LSTM model solves the problem of vanishing gradient. LSTM handles short-term and long-term dependency simply and effectively with the help of gates which are as follows

1. Forget gate: output of the previous timestamp pass to the current neuron of the observations it passes ht-1 and Xt to activation function which generate the value between 0 to 1. 0 represent as to forget the information while 1 represents as use the information.

2. Input gate: The input gate perform two operations which the main task to store the information in the cell. In the first task, information pass through the sigmoid function which updates the data based on the need. The second task is to convert information into the vector using the tanh activation function and combine both tasks into a new state.

3. Output gate: Output gate use two activation function on ht-1 and Xt with the combination of the result of activation function output gate information update the state of the information and using a combination of all gate state information, it passes to next neuron of the recurrent layer.

$$
\begin{aligned}
&\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \\
&c_t^l = f \odot c_{t-1}^l + i \odot g \\
&h_t^l = o \odot \tanh(c_t^l)
\end{aligned}
$$

In above equations,both activation function sigm and tanh are applied gate wise. Figure 3.3 demonstrate the LSTM equations.

3.2.2 Bi-Directional LSTM

A major drawback of time-series data is model difficulty to store long-term dependency of the sequence model which result in a local minimum of the loss function during training a model goes into the negative direction of the gradient and while performing

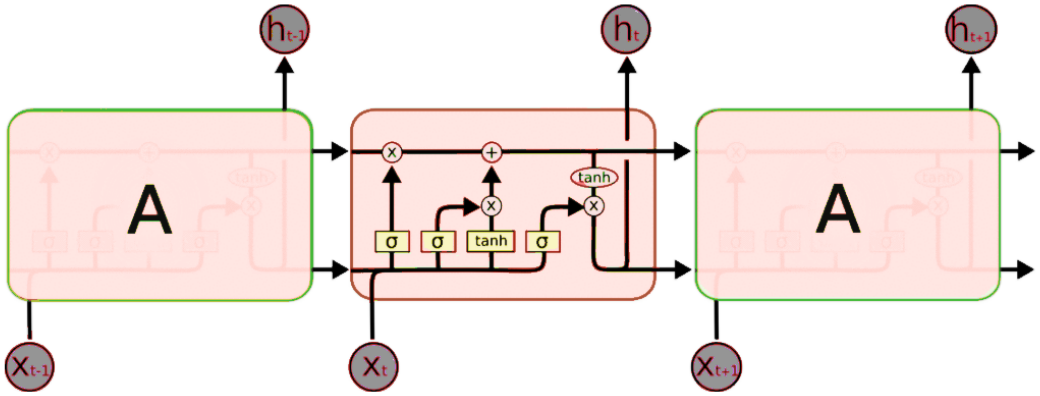


FIGURE 3.3: Long-Short Term Memory architecture

back-propagation It will multiply and iteratively become small after each operation which vanishing the gradient. As mentioned above LSTM use to solve the long-term dependency using complex architecture [19]. LSTM only uses past information to predict the future data, while Bi-LSTM has two layers of LSTM which not only use past data but also use future data. It uses both forward and backward dependency to predict the outcome, both dependencies capture varying time-series data. The following diagram shows the full architecture of the Bi-LSTM in which two LSTM one in forwarding direction from 1 ... T and second is in the backward direction from T ... 1.

1. $S_{ft} = f(A^{ft}X_t + D^f S_{ft-1} + B^f)$
2. $S_{bt} = f(A^{bt}X_t + D^b S_{bt-1} + B^b)$
3. $Y_{ft} = g(V^f S_{ft} + V^b S_{bt} + B^0)$

the above equation two layers S_{ft} and S_{bt} of LSTM both are running in opposite direction A^{bt} and D^b are the weight of backward while A^{ft} and D^f are the weight metrics of the forward LSTM network. V^f and V^b are known as output of both layers.

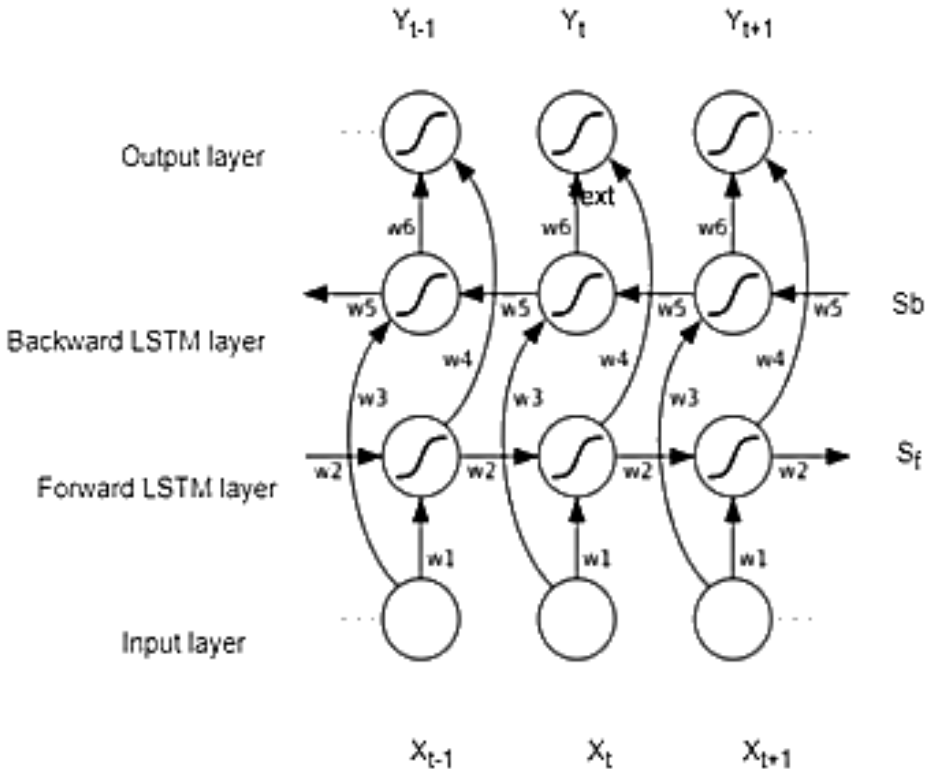


FIGURE 3.4: Bi-directional Long-Short Term Memory architecture

Chapter 4

Experimental Dataset

4.1 Dataset

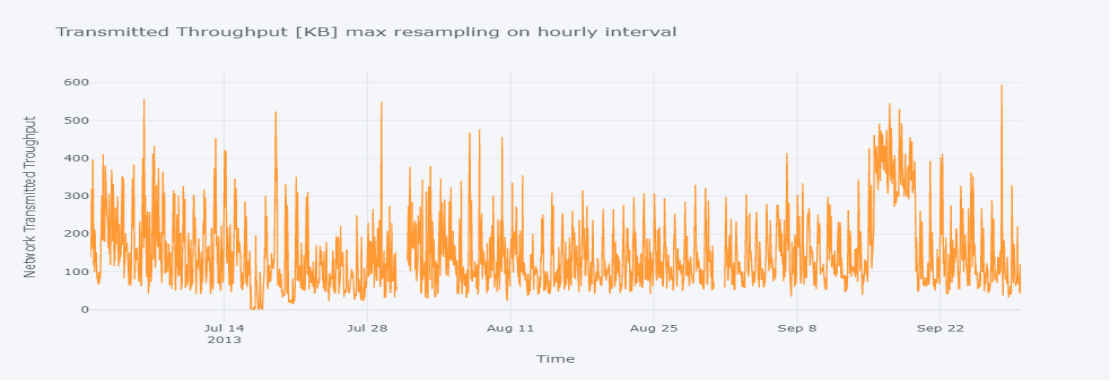
For research purposes, the GWA-T-12 BitBrain dataset[1] is used. BitBrain dataset contains the information about performance matrices of 1750 VM which are distributed on BitBrain datacenter. BitBrain Cloud provides services to maintain the hosting and compute the business enterprise operates on a VM, it has two dataset one with Fast-Storage and the second is Rnd dataset. FastStorage dataset has 1250 VM which is connected to fast storage area Network which has 1-month data with 5 min of interval. While another Rnd has 500 VM of fast storage area network and few of them are slower Network Attached Storage device, which has 3 months of data with 5 min of interval. The following performance metrics are collected by BitBrain VM.

1. Timestamp: which is in Unix format from 01-01-1970.
2. CPU capacity provisioned: which is in MHZ format.
3. CPU cores: define the number of cores in the CPU
4. CPU usage: which is measured in MHZ
5. CPU usage [%]: which is percentage format
6. Disk read throughput data is in KB/s format
7. Disk write throughput: data is in KB/s format
8. Memory usage: Memory usage measured in KB.
9. Memory capacity Provisioned: it is in a percentage format
10. Network transmitted throughput: it is in KB/s format

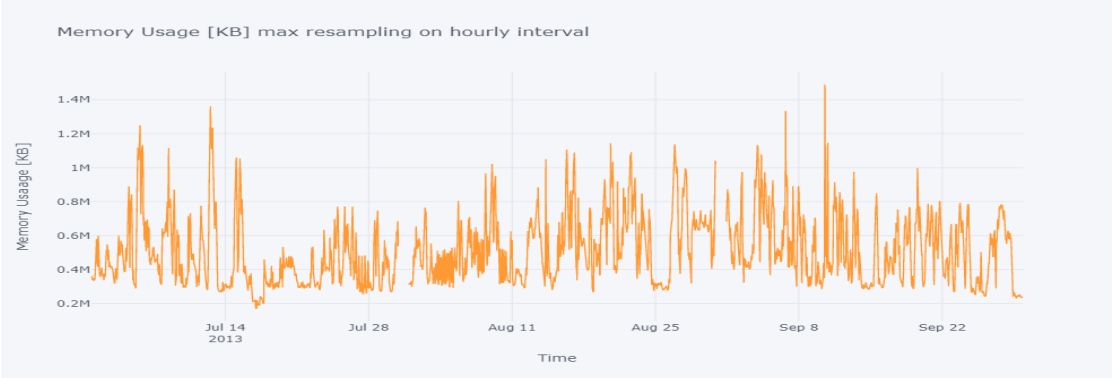
11. Network Received throughput: it is in KB/s format In this research, the Rnd dataset is more useful which has 3 months of data and more than 12 million entries of data with five minutes of interval.

4.2 Data visualization and feature engineering

Rnd Dataset has more than 12.5 million data about performance metrics. First, convert the timestamp epoch into date format to understand the time series data. Once the timestamp is in date format then use as an index of the data frame that helps to understand the flow of performance with time. With the help of features, engineering extracts the numbers of information from timestamps like identify weekdays, weekends, months, and days of capturing data. This information is very useful to find trends or seasonality in the dataset. To run model, resample data based on hourly basis which provides a frequency of data helps to predict the model and resampling sometime discover long term pattern that is very useful when forecasting data. After resampling data following figure 5 give data visualization based on an hourly basis of different performance matrices.

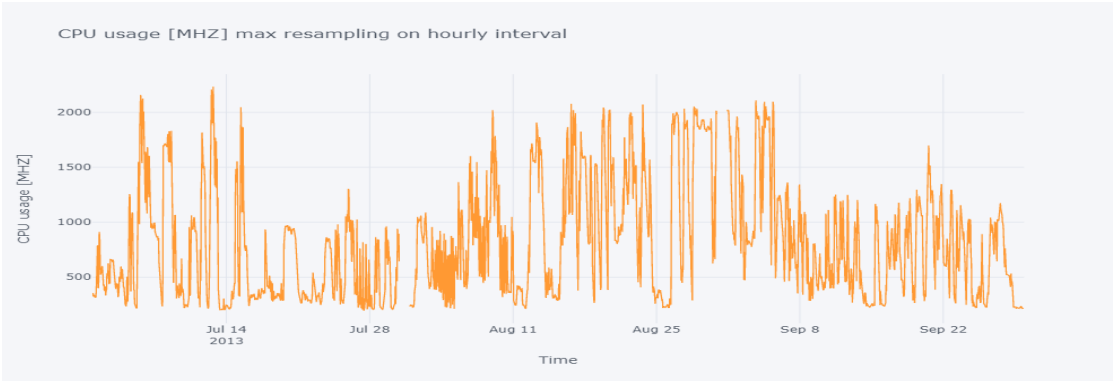


(A) Transmitted throughput

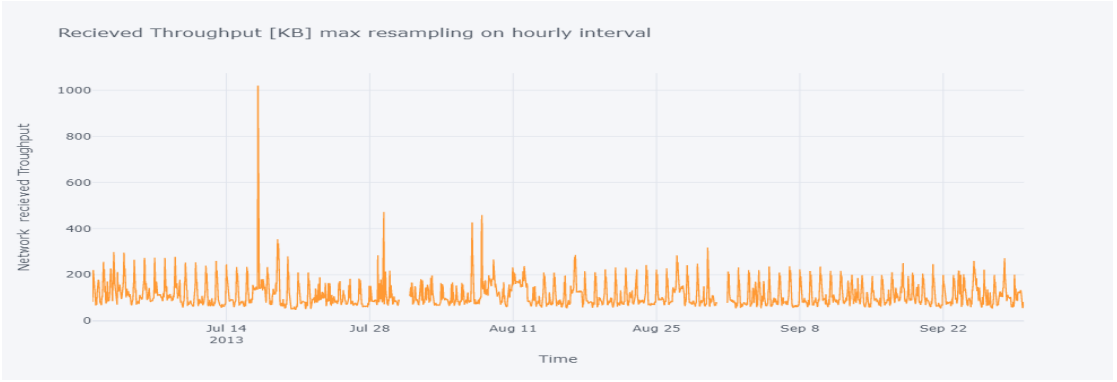


(B) Memory Usage

FIGURE 4.1: Data Visualization of performance metrics

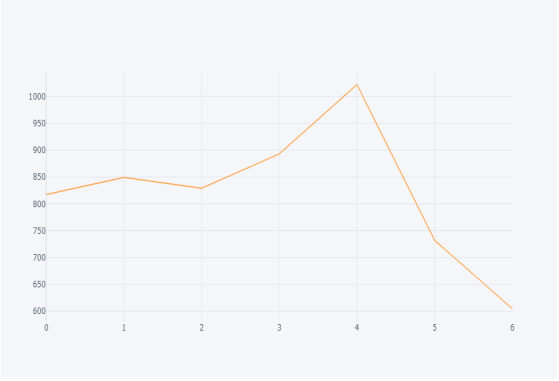


(A) CPU usage

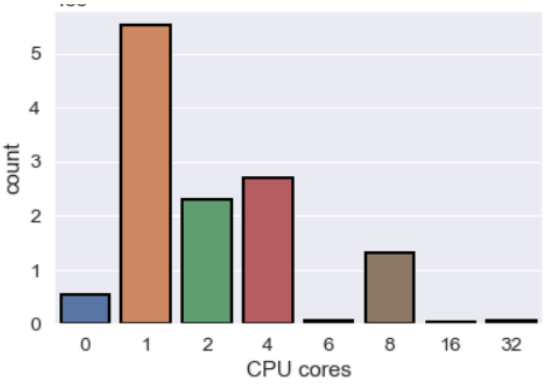


(B) Received Throughput

FIGURE 4.2: Data Visualization of performance metrics



(A)



(B)

FIGURE 4.3: CPU workload on day of week and CPU core Visualization

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|$$

FIGURE 4.4: Error metircs

Figure 4.2 is the data visualization of hourly based data which is 2184 entries . First figure includes the data visualizations of performance metrics like CPU usage, Memory Usage, Network Transmitted throughput and network received throughput. The figure 6 describe about CPU workload based on a days of the and CPU core used by VM in the dataset. In Figure 4.3(A) 0 to 6 represent Monday to Sunday day of the week , and normally Friday is the busiest day where most people use VM to perform their task while Sunday is quietest day of the week.Figure 4.3 (B) mentioned about CPU core which used in VM to perform task, 1core CPU is widely used in this dataset while 32 core CPU is rarely used to perform big task.

4.3 Error Metrics

For evaluating all models based on error we are using two error metrics in this project. First is RMSE (Root Mean Square Error) which uses the square root of the MSE metric. Another metric is MAE which is the Mean Absolute Error. Both equations are mentioned below.

Chapter 5

Time-series model

Figure 5.1 describes the decomposition of CPU usage based on trends, seasonality, levels, and residual (errors). Those attribute helps to understand time-series dataset and discover the pattern in a dataset that helps to forecast the future data. 'statsmodels.tsa.seasonal' package provides the decompose of a dataset which is mentioned in Figure 5.1 .

Data is Stationary or not?

As mentioned in chapter 3, for time series analysis data should be in stationary form. There is a number of tests are available to check data stationery or not.

1. Dicky-Fuller Test
2. KPSS test
3. The Zivot and Andrews Test
4. Variance Ratio Test

Augmented Dicky-fuller Test

Augmented Dicky-fuller test is the Unit root test which is normally used to check data is stationary or not. Unit Root test identifies all the characteristics in the time series data which makes it non-stationary. It performs an autoregression model to optimize the relevant information among different lag observations. ADF has two Hypothesis, one Null Hypothesis which derive from the unit root test and represent data is non-stationary, while the Alternate Hypothesis represents data is stationary (21). The p-value in the ADF provides an interpretation of the result, if the p-value below the threshold i.e., 5% or 1% which makes the data stationary by rejecting Null Hypothesis. On the other side, if your p-value is above the threshold i.e., (5% or 1%) which makes data

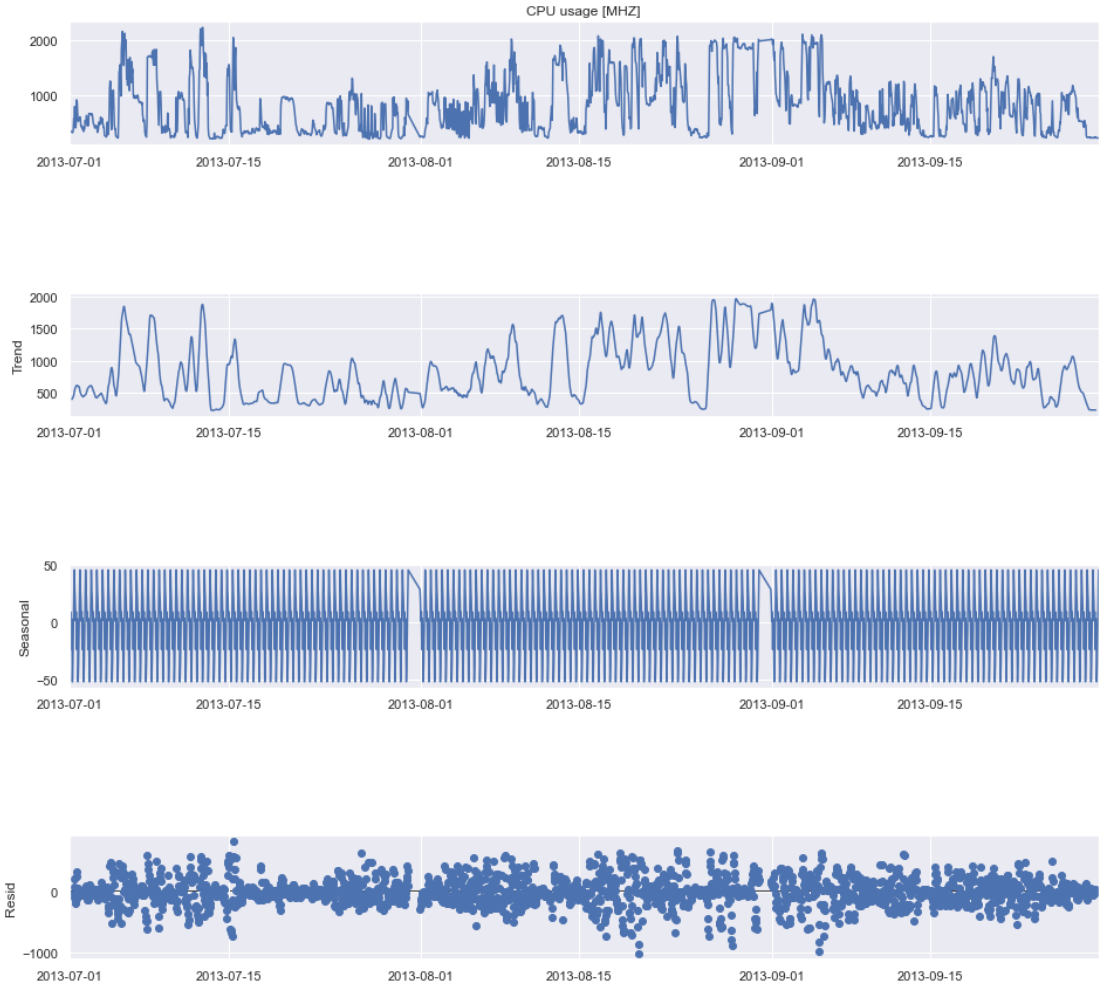


FIGURE 5.1: Decomposition of CPU usage data

Non-stationary. In python, the stats model package defines 'adfuller()' to perform the ADF test [20].After performing the ADF test on the BitBrain dataset, it concludes that attributes of a dataset (CPU usage, Memory usage, network transmitted throughput and Network received throughput) is stationary.

5.1 ARIMA Model

An ARIMA model first checks data is stationary using the ADF test which is mentioned above, after consideration above test pass through the ARIMA model. ARIMA model has 3 parameter (p, d, q). To hyper tuning parameter in the ARIMA model, there is one python package 'pmdarima' provide auto_arima function that is very useful to select the optimal value of p,d,q. they compare AIC (Akaike information criterion) and the lowest score value of AIC choose to predict the ARIMA model. AIC is an error predictor which calculates the total number of information loss by the model if the AIC score is

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=28919.939, Time=0.28 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=35438.346, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=28919.347, Time=0.08 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=33150.644, Time=0.34 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=28919.976, Time=0.15 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=28918.912, Time=0.47 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=28912.224, Time=0.84 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=28921.355, Time=0.24 sec
 ARIMA(3,0,2)(0,0,0)[0]             : AIC=inf, Time=1.54 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=28909.615, Time=0.70 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=28921.719, Time=0.37 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=28886.034, Time=0.68 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=28891.536, Time=0.27 sec
 ARIMA(3,0,3)(0,0,0)[0]             : AIC=inf, Time=1.98 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=28827.765, Time=2.54 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=28838.886, Time=0.58 sec
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=28836.266, Time=2.02 sec
 ARIMA(3,0,3)(0,0,0)[0] intercept   : AIC=28831.975, Time=1.34 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=28853.622, Time=0.46 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=28829.980, Time=2.40 sec

Best model:  ARIMA(2,0,3)(0,0,0)[0] intercept
Total fit time: 17.367 seconds
```

FIGURE 5.2: AIC score of auto_arima function

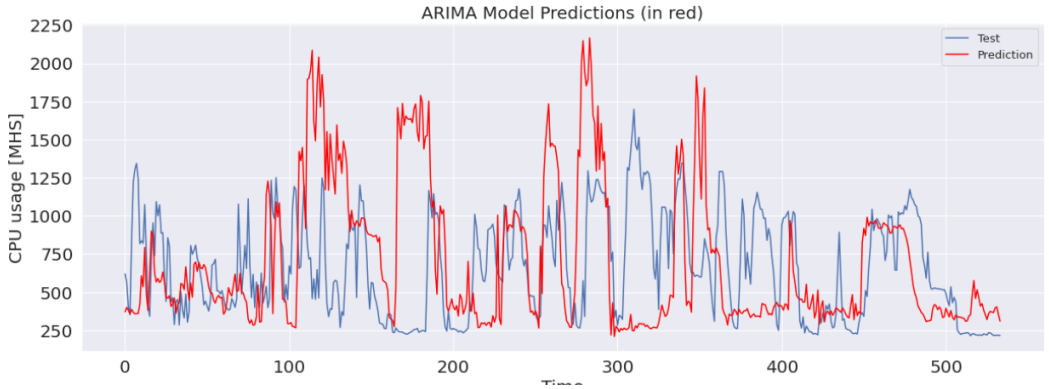


FIGURE 5.3: Test vs Prediction model on ARIMA(2,0,3)

less then model is highly qualified for forecasting the result. Figure 5.2 mentioned above all possible combinations that perform on the ARIMA model and based on the AIC score choose optimal value of p,d,q. based on the result ARIMA model predict based on (2,0,3). Figure 5.3 describes the result of ARIMA(2,0,3) model prediction with test data and based on error metrics RMSE error is 537.6187 and MAE score is 386.8786. good model RMSE score is less than 180 so the ARIMA model not effective to predict a good result.

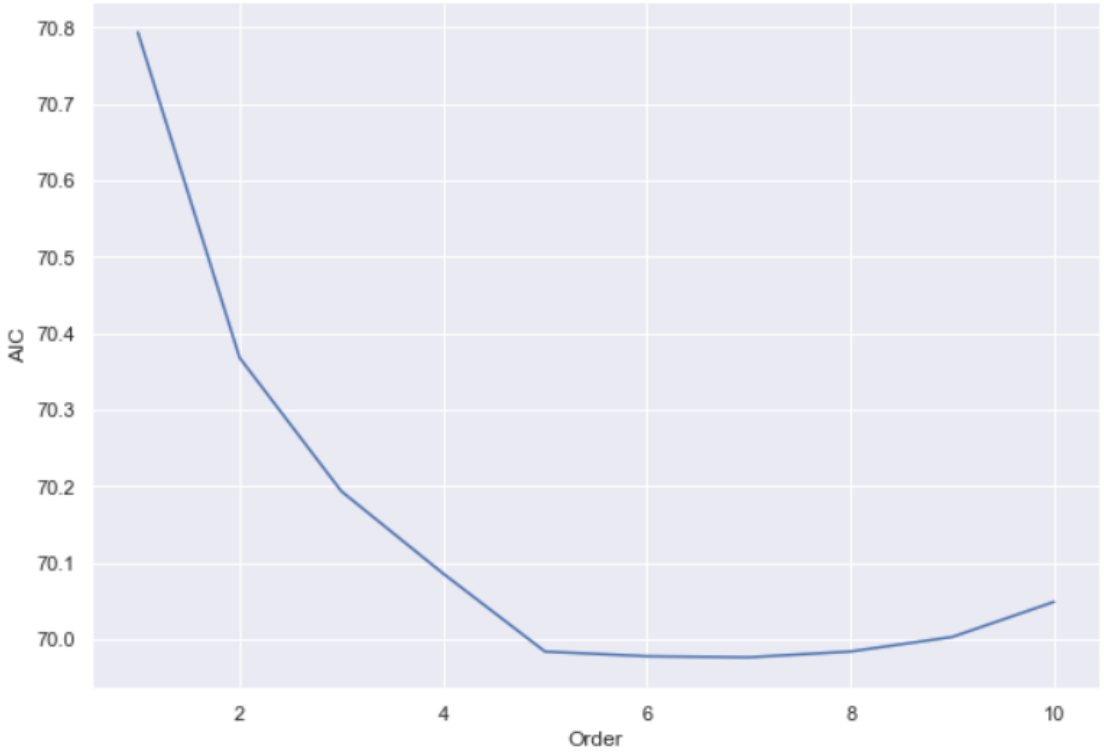


FIGURE 5.4: lag order plot based on AIC score

5.2 Vector Auto Regression(VAR) Model

VAR model normally used to predict multivariate time series, the dataset is divided into two parts 75% as training data and remaining for testing a model. Before applying data to the model, first check data is stationary or not. VAR Model uses Augmented Dicky-Fuller(ADF) test to check the data is stationary or not. Parameter of dataset i.e. CPU usage, Memory Usage, Network Transmit throughput, Network received Throughput, is all stationary.

TABLE 5.1: AIC score for lag values

lag order	AIC score
1	70.7932
2	70.3678
3	70.1933
4	70.0852
5	69.9839
6	69.9778
7	69.9764
8	69.9839
9	70.0030
10	70.0489

VAR model based on Auto Regression so to get optimal value of lag value, try max value of lag as a 10 and calculate AIC score which mentioned in 5.1. based on the lowest score of AIC select the lag value and pass it to the model. Figure 5.4 represents the AIC score and plot for the VAR model based on comparison select the lowest AIC score lag value to train the model. Using 7 lag value for p VAR model trained and predict the CPU cores with 0.2287 MAE error and 0.3538 RMSE error, for CPU usage 401.026 MAE error and 479.814 RMSE error, for memory usage 1633.6453 MAE and 1915.21 RMSE, for disk read throughput prediction error 34.775 MAE and 43.2363 RMSE, and, network received throughput error is 22.713 MAE and 33.23 RMSE with respect to BitBrain dataset.

Chapter 6

Neuaral Network model

6.1 LSTM model

In this section we present the experimental results based on research studies on BitBrain data set. For training a model BitBrain Rnd dataset is very useful. Dataset has 3 month trace of 5 min of interval. More than 12 million dataset of 5 min of interval trace resample into hourly based to predict the future resoruce usage on VM. From dataset 75 Figure 6.1 mentioned how to predict 1-hour data based on previous 6h information where previous 6h data classified as input data for current timestamp data. After convert input data and

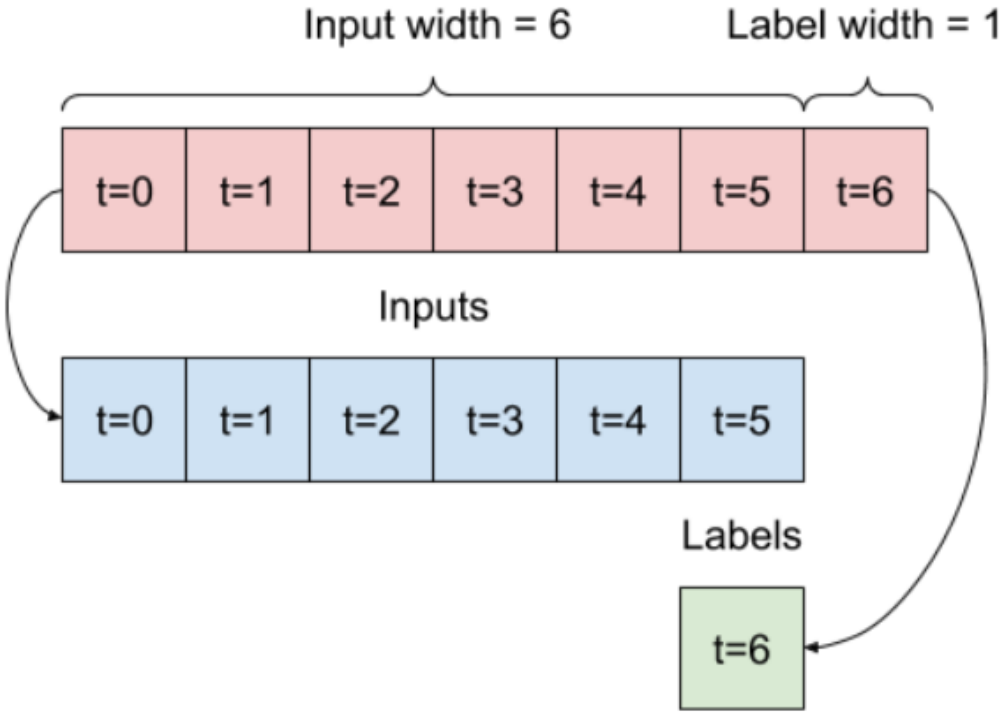
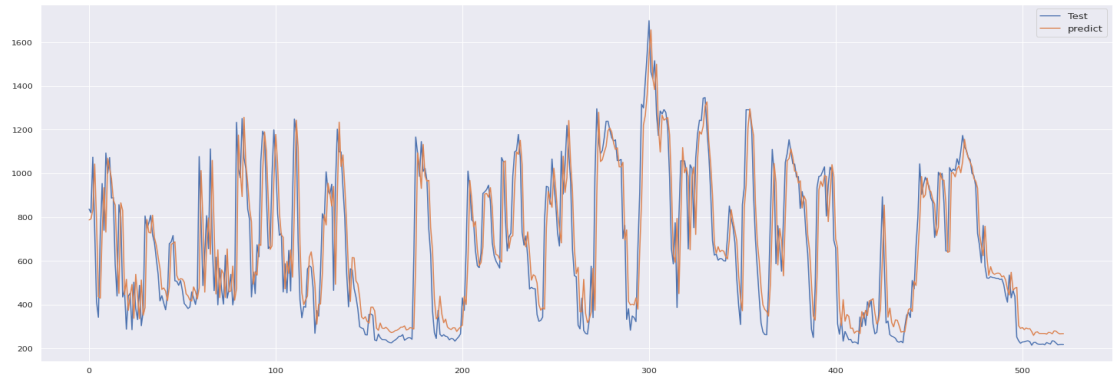
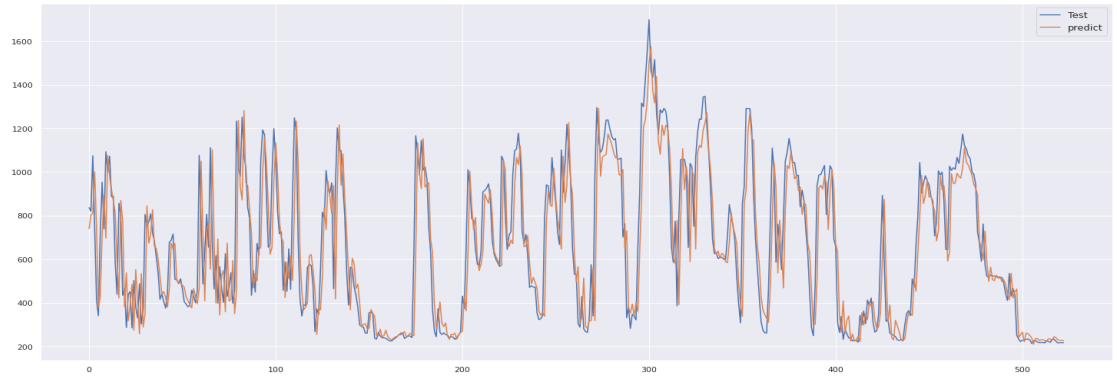


Figure 6.1: lag order plot based on AIC score

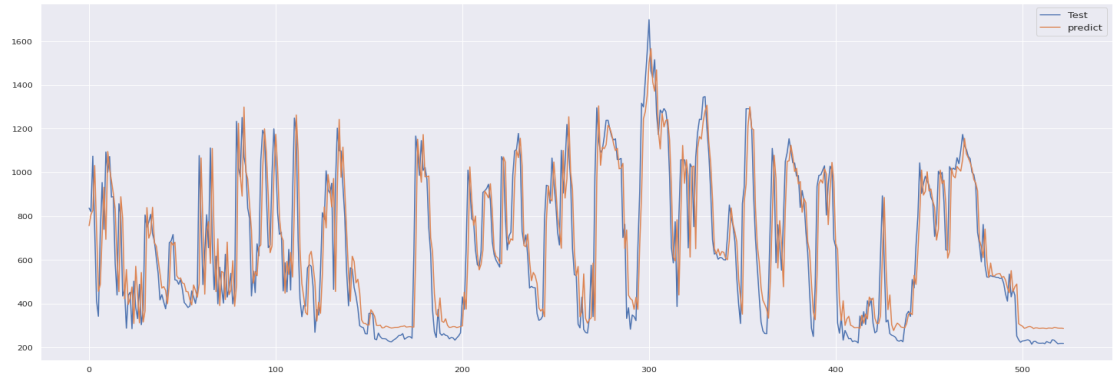
labels it is important that data must be in (samples, timestamp, feature) format.Reshape dataset and pass the information to the all sequential model. For experimental analysis dataset pass to four model first vanilla LSTM, multi hidden layer LSTM, first Bi-LSTM, and multi-hidden layer LSTM.



(A) Vanilla LSTM



(B) Vanilla LSTM model with lambda layer



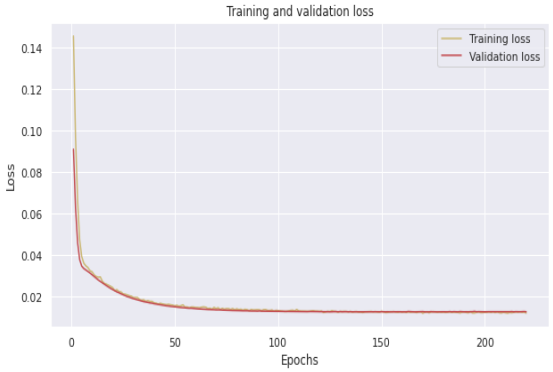
(C) Multilayer of LSTM model

FIGURE 6.2: test vs prediction of data on LSTM Model

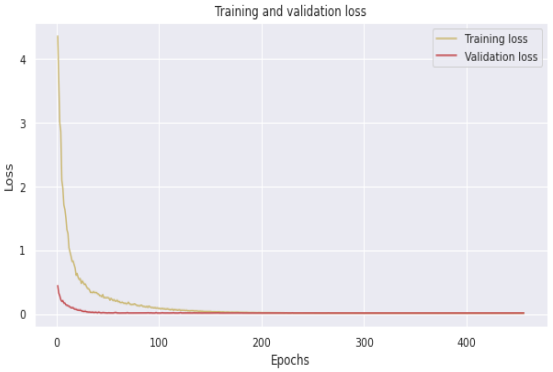
All the vanilla models use 90 units for LSTM layers with ReLU activation function after that use the dropout layer to prevent overfitting problems, then add a dense layer to the model. After creating a model compile model in which we use adam optimizer and also fetch accuracy and make metrics for the model. For training a model we used 32

batch sizes with 500 epochs in every Vanilla-based LSTM and Bi-LSTM model. Multiple hidden layers of LSTM and LSTM use 3 hidden layers with 90 units for each LSTM layer. The model uses ReLU activation with a 0.3 value dropout layer to prevent overfitting problems. Both models use adam optimizer with accuracy and MAE metrics, also use 32 batch size with 500 epochs to train a model.

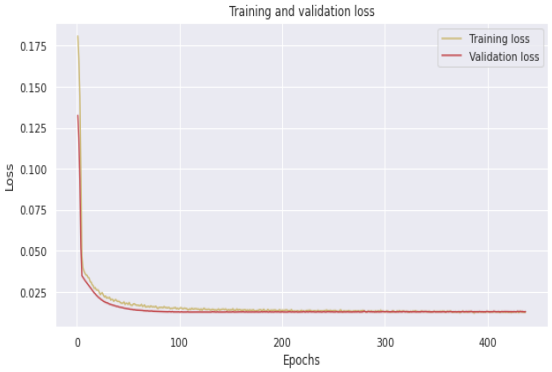
Figure 6.2 describes information about performance on a variant of the LSTM model. Figure 6.2(A) describes the Vanilla LSTM model, after training a model its generates 158.26 RMSE and 105.10 MAE error on training data while 166.83 RMSE and 119.42 MAE error on testing data. Figure 6.2(B) describes the LSTM model with the Lambda layer which uses $x = x * 100$ comparison of data of the dense layer. It produces 158.55 RMSE and 101.81 MAE errors on training data while 167.74 RMSE and 114.95 MAE errors on testing data. Figure 6.2(c) describes multiple layers of the LSTM model with 156.52 52 RMSE error and 102.83 MAE error on train data for testing data it generates 168.12 RMSE and 121.01 MAE error. Figure 6.3 describes a comparison of training loss with validation loss by each epoch while training a model



(A) Vanilla LSTM



(B) Vanilla LSTM model with lambda layer



(C) Multilayer of LSTM model

FIGURE 6.3: loss of the model

6.2 BI-LSTM model

Figure 6.4 describe different Bi-LSTM model with single and multiple layers of Bi-LSTM. Figure 6.4 (a) describe the single layer of Bi-LSTM which produce 154.22 RMSE and 101.35 MAE.Error while training a model and 166.16 RMSE and 119.59 MAE errors while testing data applied on the model. It produces good prediction when CPU Usage to low but there is a spike in CPU usage then the prediction is not accurate to forecast the future workload. figure 6.4 (b) produces Multilayer Bi-directional LSTM on the CPU usage dataset. there is an improvement in error while training a model, it has 143.23 RMSE and 93. 49 MAE while multilayer Bi-LSTM does not produce an accurate prediction on testing data, it has 170.76 RMSE and 123.23 MAE errors. Multiple layers of Bi-LSTM produce the best prediction when there is a spike in CPU usage but do not produce an accurate result when CPU usage is very low. Figure 6.5 describe training loss and validation loss based on the epoch of both Bi-LSTM model. All

(A) single layer Bi-LSTM

(B) multilayer Bi-LSTM model

FIGURE 6.4: test vs prediction of data on Bi-LSTM Model

the models use the early stopping method, early stopping is a method that uses to prevent overfitting the training model. Early stopping stops the training model on a large epoch if the performance of the model stops improving on validation observations [21].Table 6.1 compare all deep learning model on training and testing data while applying model

TABLE 6.1: comparative analysis of training and testing error with Deep Learning model

model	train		test	
	RMSE	MAE	RMSE	MSE
Vanilla LSTM	158.26	105.10	166.83	119.42
LSTM with Lambda Layer	158.55	101.81	167.74	114.95
Multilayer LSTM	156.52	102.83	168.12	121.01
Single Bi-LSTM	154.22	101.35	166.16	1119.59
MultiLayer LSTM	143.23	93.49	170.76	125.25

on a training data Multilayer LSTM produce efficient while applying same model on a testing data Single Bi-LSTM is effective one. result Other performance metrics applied on all models which are mentioned above and for all the predictions refer to Appendix A.

(A) single layer Bi-LSTM

(B) multilayer Bi-LSTM model

FIGURE 6.5: loss of the model

Chapter 7

Future work

While performing a comparative analysis of a model, the Bi-LSTM model produces efficient prediction compare to other models that modelslanning to improve the model and produce more accurate results with the help of a more complex model and using a transformer. There are other complex models like CNN-LSTM and Conv-LSTM may produce good result compare to Bi-LSTM, also using Bi-Directional LSTM encoder-decoder we can reduce the error on training and testing data.

We are planning to apply the same model on a cloud platform like AWS, google cloud, Azure. First, collect metric information of VM on which your application is running, once we collect all the data apply those data to Bi-directional LSTM using machine learning service i.e. for AWS there is on service Amazon Sage Maker which helps to run the model on AWS and integrate service with your application based on the real-time load of CPU usage it predict the future workload for small period i.e 2 to 5 hr and based on prediction adjust the scheduling of VM. If CPU load is too low then it will freeze the VM or model predict the spike in future CPU usage it provisioned some VM in advance to save the time of provision resources when there is a spike of CPU usage in the applications. When you train to predict the other performance metrics of the model then you can save lots of resources that are not used by the application and using resource provisioned in advance it may save your time when load increase.

Chapter 8

Conclusion

Massive usage of Cloud computing, it is important that not only provide services without any delay which are implemented on a cloud but also produce the effective resource utilization method for all the resources which are used by that services. We proposed a Bi-Directional LSTM model to predict resource usage on a BitBrain cloud dataset. the proposed model more effective to predict resource usage like CPU, memory, network on VM machines. We also perform classical time series models like ARIMA and VAR to understand the performance produced by those model. And compare with a different variants of the LSTM and Bi-LSTM model. There are some very interesting implementations and application of the proposed model in the field of cloud environments which helps to improve the strategy of resource allocation on a cloud.

List of Figures

List of Tables

Bibliography

[1] Siqi Shen, Vincent Van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 465–474. IEEE, 2015.

[2] Yizeng Chen, Xingui Li, and Fangning Chen. Overview and analysis of cloud computing research and application. In *2011 International Conference on E-Business and E-Government (ICEE)*, pages 1–4. IEEE, 2011.

[3] Anagha Yadav and SB Rathod. Priority based task scheduling by mapping conflict-free resources and optimized workload utilization in cloud computing. In *2016 International Conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE, 2016.

[4] ABM Bodrul Alam, Mohammad Zulkernine, and Anwar Haque. A reliability-based resource allocation approach for cloud computing. In *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*, pages 249–252. IEEE, 2017.

[5] Martin Duggan, Karl Mason, Jim Duggan, Enda Howley, and Enda Barrett. Predicting host cpu utilization in cloud computing using recurrent neural networks. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 67–72. IEEE, 2017.

[6] Eduard Zharikov, Sergii Telenyk, and Petro Bidyuk. Adaptive workload forecasting in cloud data centers. *Journal of Grid Computing*, 18(1):149–168, 2020.

[7] Jitendra Kumar and Ashutosh Kumar Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41–52, 2018.

[8] Peter A Dinda and David R O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, 2000.

[9] Parminder Singh, Pooja Gupta, and Kiran Jyoti. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, 22 (2):619–633, 2019.

[10] Carlos Vazquez, Ram Krishnan, and Eugene John. Time series forecasting of cloud data center workloads for dynamic resource provisioning. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 6(3):87–110, 2015.

[11] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[12] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao. A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–18, 2019.

[13] Shaifu Gupta and Dileep Aroor 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)*, pages 1–6. IEEE, 2017.

[14] Soukaina Ouhame, Youssef Hadi, and Arif Ullah. An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. *Neural Computing and Applications*, pages 1–13, 2021.

[15] Hengheng Shen and Xuehai Hong. Host load prediction with bi-directional long short-term memory in cloud computing. *arXiv preprint arXiv:2007.15582*, 2020.

[16] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, pages 1–17, 2015.

[17] Deepak Janardhanan and Enda Barrett. Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 55–60. IEEE, 2017.

[18] Mathias Lechner and Ramin Hasani. Learning long-term dependencies in irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

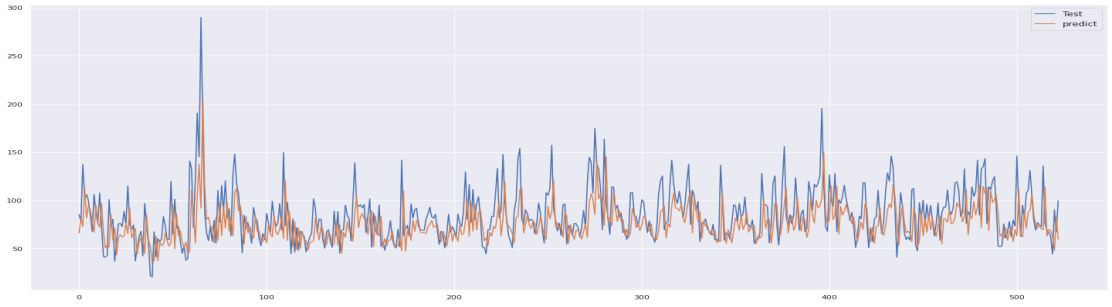
[20] Yin-Wong Cheung and Kon S Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

[21] Grace Guan and Barbara E Engelhardt. Predicting sick patient volume in a pediatric outpatient setting using time series analysis. In *Machine Learning for Healthcare Conference*, pages 271–287. PMLR, 2019.

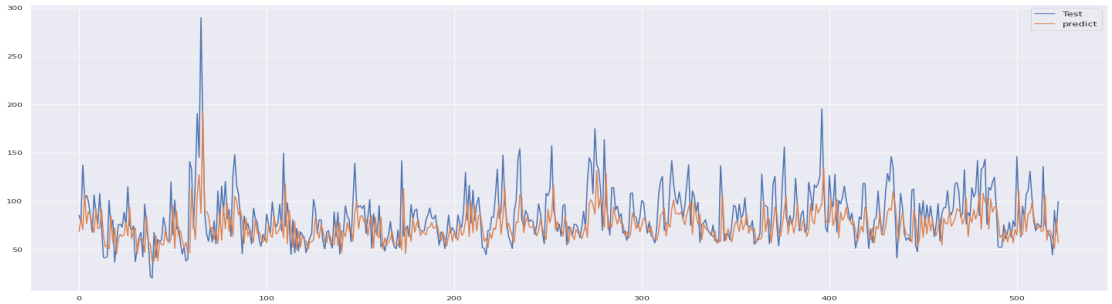
Appendix A

MODEL RUN WITH DIFFERENT RESOURCES

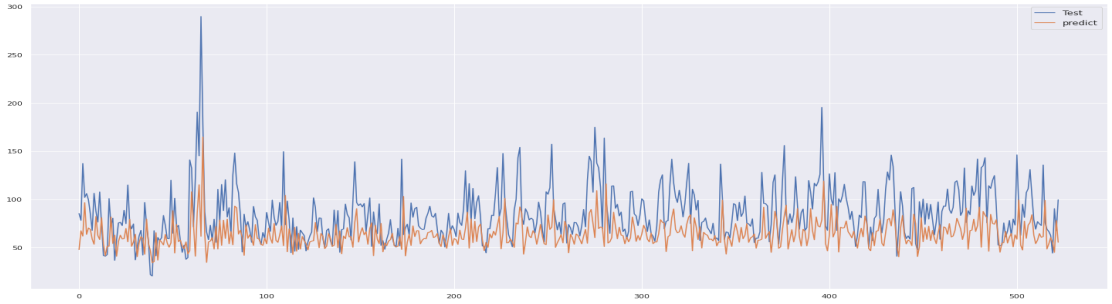
In Appendix A we are using Network received throughput performance metrics data as an input dataset and pass through all Neural Network model that is defined in Chapter 6. First resample data based on hourly basis to forecast short time prediction. after that divide 75 % data as a training data and remaining are used as a testing data. normalize operation perform and produce normalize result into 0 to 1 scale. After that create input data and label based on previous lag value informations.After creating creating data pass information to the vanilla LSTM model, Figure A.1 (A) describes Vanilla LSTM model, after training a model it produces 12.63 RMSE and 7.53 MAE error on training data while 27.73 RMSE and 19.99 MAE on testing data. Figure A.1(B) describes Vanilla based LSTM model with lambda layers which has 12.50 RMSE and 7.53 MAE error on training data, while running a testing data on a model 27.80 RMSE and 19.94 MAE error. Figure A.1 (c) is multilayer LSTM model which has 12.28 RMSE and 7.15 MAE error on training data, 33.84 RMSE and 24.83 MAE error on testing data. we also applied data to Bi-LSTM model with single layers which produce 11.61 RMSE and 6.86 MAE error on training data while 34.69 RMSE and 25.37 MAE error on testing data.final Bi-LSTM with multilayer produced 11.50 RMSE and 6.77 MAE error on training data while 35.78 RMSE and 26.37 MAE error in testing data. As you observe that model produce very efficient result on training data while running a model on a testing data Vanilla LSTM model is pretty good compare to other model.



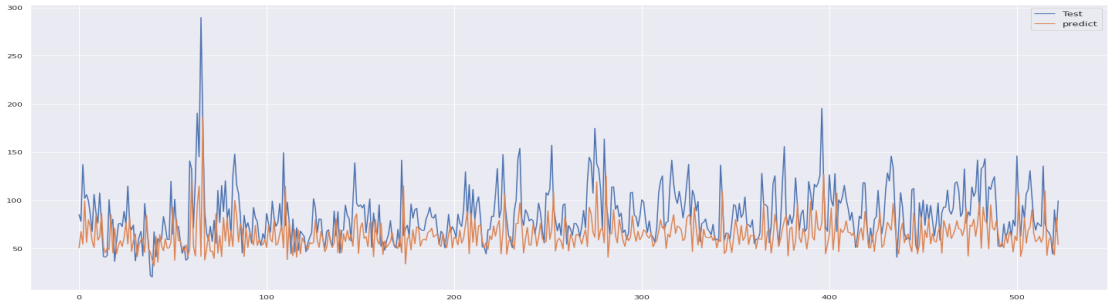
(A) Vanilla LSTM



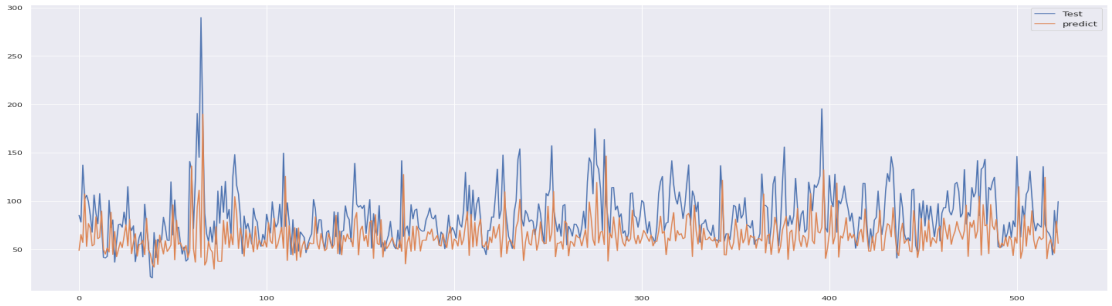
(B) Vanilla LSTM model with lambda layer



(C) Multilayer of LSTM model



(D) single layer Bi-LSTM



(E) multilayer Bi-LSTM model

FIGURE A.1: test vs prediction of data on All Model