Cloud Workload Prediction

A Project Report Submitted
in Partial Fulfillment of Requirements
for the Degree of

Bachelor of Technology

by

Harsh Srova (2018CSB1095)
Guzzarlapudi Stephen Sugun (2018CSB1225)

Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar 140001, India
April 2022

Abstract

Cloud is the foremost demand of the current century. The market for the cloud is increasing at a rapid rate. Studies suggest that the market will grow from current valuation of $ 445 billion to $ 947 billion by 2026. That translates to a CAGR of 16.3%. Cloud technology is helping enterprises to grow their business. With an increase in the number of users, enterprises are using the cloud to store, manage and process critical data. The current leading technologies, such as data science, and machine learning, are also pushing for the cloud. The data that is stored in the cloud can be used as fuel for the machine learning algorithms. During covid 19, many enterprises initiated the WFH. Due to this Microsoft Team platform saw a rise of 44 million users globally due to the high demand for collaboration solutions. Zoom platform saw a growth of 326% users, and the profit jumped from $ 21.7m in 2019 to $ 671.5m. The demand for cloud is increasing at an unprecedented rate. With more and more users coming onto the cloud, we need a very flexible dynamic cloud. A dynamic cloud changes the number of resources and automatically provisions and adjusts itself as a cloud workload changes. Therefore we need an efficient workload prediction method. This paper addressed the cloud server workload prediction using the transformer model. We used the NASA HTTP server log dataset to train our model. The best accuracy we got is a mean square error of 2.95 x $10^{-}3$.

Acknowledgements

We would like to express our gratitude to our project supervisor Dr Shweta Jain of the Department of Computer Science and Engineering at the Indian Institute of Technology, Ropar. She helped us throughout the course of this project and gave us guidance whenever necessary.

We would also like to acknowledge Dr Puneet goyal of the Department of Computer Science and Engineering at the Indian Institute of Technology, Ropar for helping us and informing us about the needed requirements of the project.

Thank you

Honor Code

We certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Harsh Srova (2018CSB1095)

Guzzarlapudi Stephen Sugun (2018CSB1225)

Certificate

It is certified that the B. Tech. project "Cloud Workload Prediction" has been done by Guzzarlapudi Stephen Sugun (2018CSB1225), Harsh Srova (2018CSB1095) under my supervision. This report has been submitted towards partial fulfillment of Capstone project (CP303) requirements.

Dr. Shweta Jain
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud Systems have become very prominent in recent years. Since the inception of the internet, as more and more people regularly use the internet, server size has increased exponentially. Many applications and protocols need cloud servers to function. The COVID-19 pandemic and the subsequent lockdowns forced many people into their houses. This resulted in many businesses and companies enact work from home policies. This, in turn, increased demand for online video conference applications, online programming terminals, online business analytic discussion programs, and many more.

All of these require quick and efficient cloud servers. This rise in demand for companies like video conference have met collaborative cloud services applications like Zoom, Google meet, Webex by Cisco, and Microsoft teams. Collaborative coding websites like Google Colaboratory. Collaborative document writing with the help of google docs. These collaborative cloud services have seen unprecedented growth during the pandemic.

The pandemic has shown that a lot of jobs can be done from home. Despite the end of the pandemic, many companies have decided to continue their work from home policies. This is due to the fact that working from home can help both the employer and the employee. The employer does not need to rent office space . Therefore, they can improve their margins. Whereas the employees may prefer to work from the comfort of their homes and would prefer not to waste

time in unnecessary commuting. Therefore, as this trend of decentralized working continues, the demand for collaborative cloud services will keep increasing.

With the number of users coming onto the cloud systems increasing, we need a flexible and dynamic cloud. A dynamic cloud alters the number of resources and automatically regulates and adjusts itself as the cloud workload changes. Therefore we need an efficient workload prediction algorithm.

1.1 Motivation

Workload prediction is important for Cloud systems. The two main reasons are dynamic resource scaling and efficient energy management.

1. Dynamic Resource Scaling: A Cloud server provides service to multiple users. The Quality of Service provided must be stable and standardized. different users have different needs, and different times of day have different workloads; therefore, we must allocate cloud resources accordingly.

2. Efficient energy management: It helps to reduce costs and makes it more economical and affordable to users. Reduced power consumption also helps the environment.

1.2 Existing Work and their Drawbacks

A great deal of research has been done looking into multiple methods to predict future workloads, namely:
1. Autoregression (AR)
2. Moving average (MA)
3. Exponential smoothing (ES)
4. Autoregressive integrated moving average (ARIMA)
5. Long Short Term Memory (LSTM)

LSTMs are deep learning models, and previously, they have been the best model

for predicting the future. However, the main issue facing them is the vanishing gradient problem, i.e. in deep learning techniques, when the network is too deep (the number of layers is very high), the network cannot propagate information without loss. This results in increasing errors in predicting the future.

1.3 Solution

To combat this problem, Transformers have been proposed. Transformers are deep learning models that use self-attention and consider each part of the input data. Research on Transformers has been done, but it is an active field of research. Transformers have been previously used for Natural Language Processing and computer vision. This paper uses Transformers to predict future workload in cloud servers, specifically the NASA HTTP logs.

In this paper, we also compare transformers with LSTMs, to see which is more efficient at predicting the future workload of cloud servers.

Chapter 2

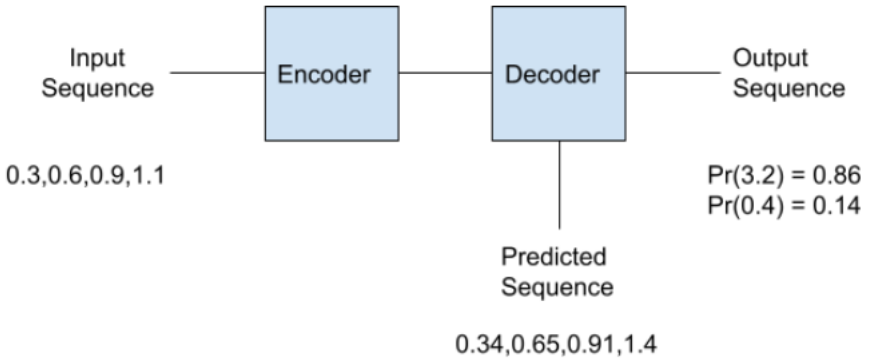
Proposed Solution

2.1 Overview

To combat this problem, we used the transformer model. Transformer models are the current state of the art in time-series prediction. We used the vanilla transformer architecture proposed in "Attention is all you need paper". Transformers are a stack of encoder and decoder blocks. Encoder and decoder blocks are further made up of self-attention blocks. Self-attention is the powerhouse of the transformers. Self-attention is made up of queries, keys and value matrices. Queries and keys interact with each other to give the values. Multi-head self-attention are h parallel self-attention blocks stacked upon each other. The outputs of these self-attention blocks are concatenated to give the final result. Figure 2.1 shows the vanilla transformer architecture. Before giving input to the transformer, we first transformed the input and added the sin-cos positional encoding.

Language	Frameworks/Libraries
Python	Pytorch Matplotlib Numpy Pandas

Table 2.1: Tech Stack



The above figure is the basic architecture of transformers. An input sequence is fed into the encoder block, and the output of the encoder is fed into the decoder along with the previously predicted output. The decoder outputs the probabilities for the workload in the next timestep. At first, the encoder is fed 0.3, and the decoder predicts 0.34. In the next step, the encoder is fed 0.3, the decoder is fed 0.34, and the decoder outputs 0.65 and so on. At last, the model is given 0.3, 0,6, 0.9 and 1.1, the decoder is given 0.34, 0.65, 0.91 and 1.4, and the model outputs 0.86 as the workload for the next timestep.

2.2 Technology Stack

Python: Python is the most famous language for machine learning tasks. Python provides various libraries such as NumPy. **Numpy** provide support for various mathematical functions to work.

Pytorch: It is an open-source machine learning framework used for computer vision and language processing tasks. We used the transformers PyTorch library for our implementation.

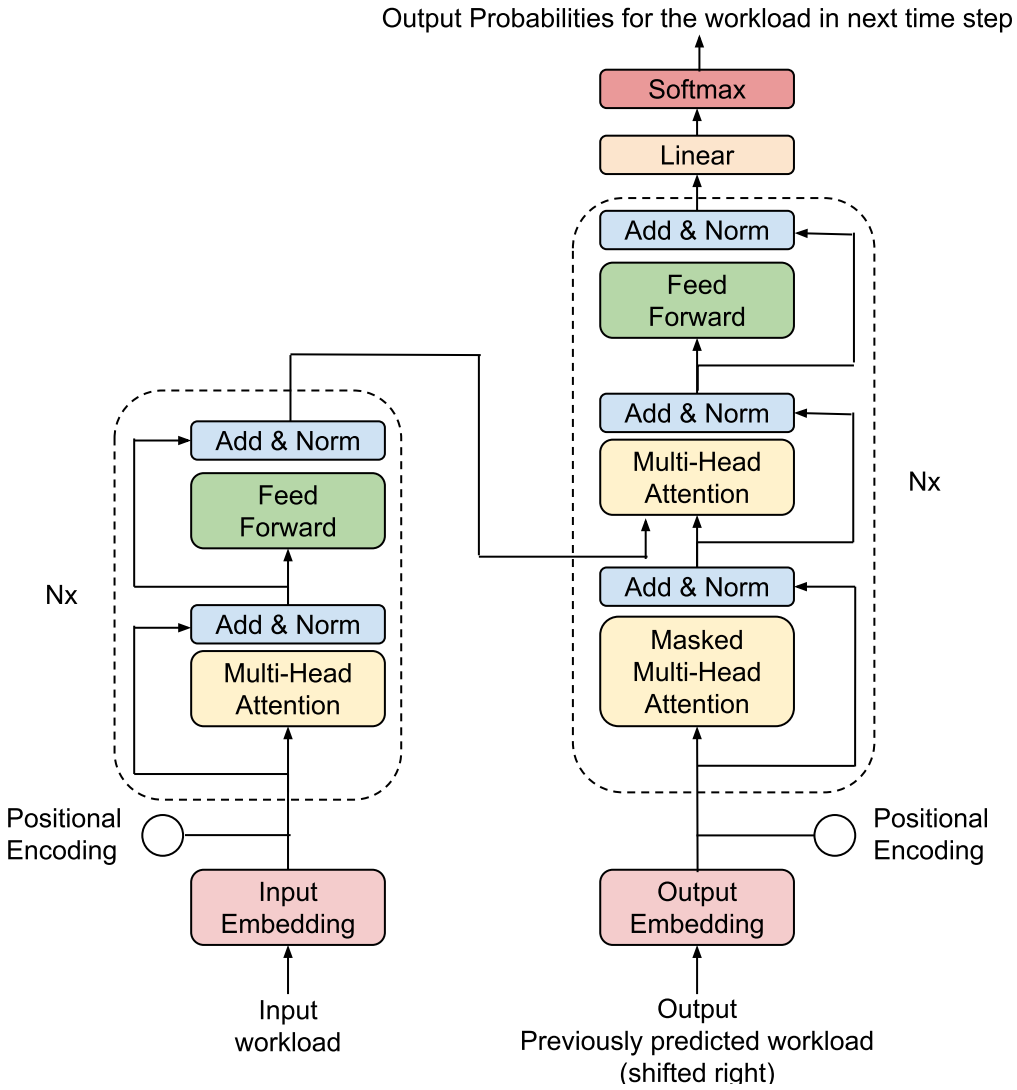


Figure 2.1: Vanilla transformer architecture purposed in "Attention in all you need" paper

Matplotlib: Matlpot is a library usually used to display results.

Pandas: Pandas is a data processing tool. We used pandas for data pre-processing.

2.3 Dataset:

There are many datasets available on the internet for this problem. We used the NASA HTTP server log dataset. Dataset has six attributes Host, Time, Method, Response, URL and Bytes. It has 3.26 million data points before pre-processing.

We did the pre-processing on the dataset. Pre-processing is done to extract useful information from the dataset. We added the load attribute, where the load is the number of users arriving per unit time. We divided the dataset into two datasets.

1. Seconds Dataset

2. Minutes Dataset

In the second dataset, unit time is second, and in the minute dataset, the unit time is minute. Figure 2.2 shows the dataset for the minutes, and Figure 2.3 shows the dataset for the seconds. We can already see some patterns emerging in the graphs. Our goal is to extract this pattern using the tranformer model.

2.4 Experiments:

We compared results of our experiment to LSTM model. The values for LSTM are taken from a published paper "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters".

Settings: We used the google colab for training our model. Google colb uses GPU of 1xTesla K80. It have 2496 cuda cores and 12 GB of GDDR5 VRAM, CPU of single threaded Xeon processors @ 2.3 Ghz and disk of 33 GB.

Dataset: We used 60% of dataset for training and 40 % for testing.

Hyperparameters: We experimented with different batch sizes and our model performed best on batch size of 50 and learning rate 0f 0.005. We took the

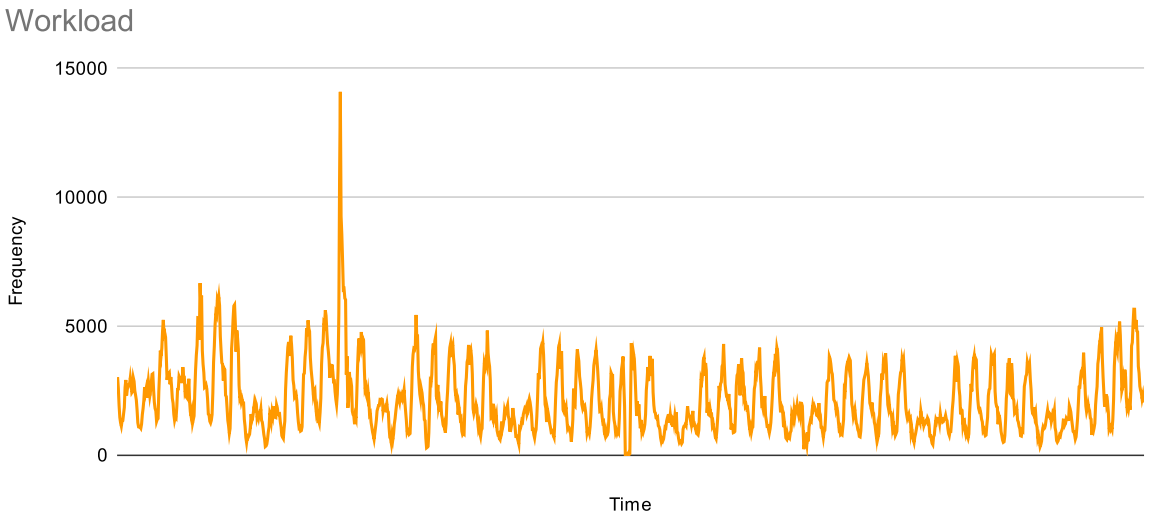


Figure 2.2: Minute Workload

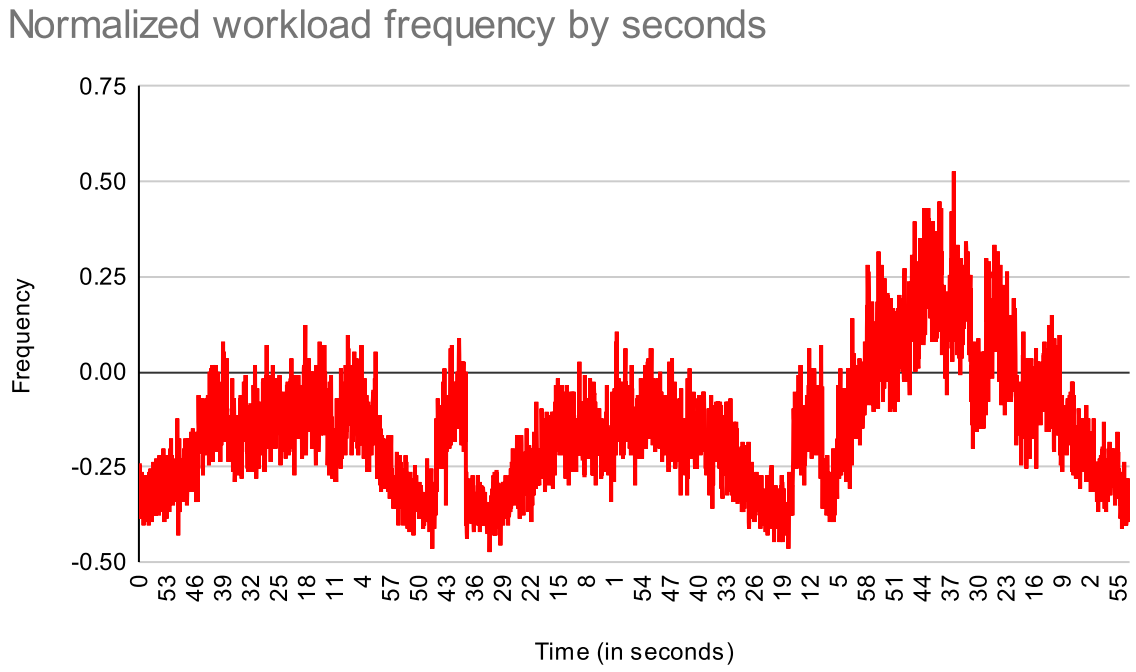


Figure 2.3: Seconds Workload

TWS as 100. TWS is number of previous timestep workload used to predict the workload for future timestep in training. We varied the PWS (1, 5, 10, 20, 30 and 60) to comapre the results with LSTM.PWS is number of previos timestep workload taken to predict future workload.

Loss Function: We used the mean squared loss function. Each token contribute equally to the loss.

Chapter 3

Improvements to Existing model

We have improved the existing model in two main areas: The speed of computation and accuracy. Our model is faster than the previous model, and our model has increased accuracy. This was achieved by using Transformers. Transformers are state of the art for time-series prediction. Therefore the accuracy is very high, and the error is low. Transformers harness the power of multiprocessing by working in parallel. In the Multi-head self-attention, different self-attention blocks run on different cores. Therefore training and processing speed are fast.

3.1 Increased accuracy

This was achieved by optimizing and fine-tuning hyperparameters such as batch size. Batch size is a hyper-parameter representing the number of inputs that the model goes through before updating internal model parameters. We tested for multiple batch sizes to find the optimal one. We also normalized the dataset to be used with more ease and efficiency. Cyclic temporal modelling was applied to the dataset to find inherent cyclical and periodic behaviours in the dataset. This helps the algorithm to identify better long term trends, which in turn translates to better accuracy.

3.2 Increased speed

This was achieved by reducing the number of layers. We found that increasing the number of layers drastically increases the training and processing time. This happens because of the vanishing gradient problem, i.e. as the number of layers increases, early information is lost. Therefore we reduced the number of layers.

Chapter 4

Results

4.1 Mean squared errors of different prediction models

Comparison Results			
PWS	Transformer(s)	Transformers(m)	Lstm
1	3.68	4.47	13.06
5	2.66	3.5	4.79
10	2.29	3.52	6.66
20	2.15	3.12	7.01
30	2.05	3.01	6.43
60	1.89	2.95	5.59

Table 4.1: Comparing Mean squared error of Transformers and LSTM models (results are x$10^{-}3$)

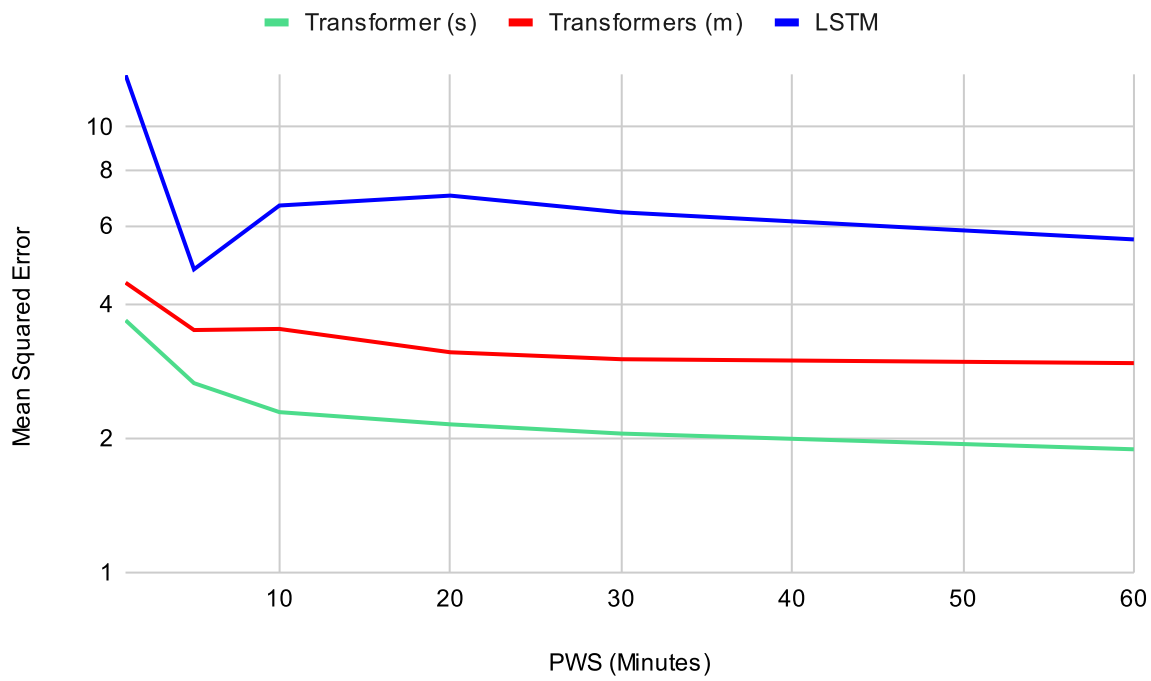


Figure 4.1: Mean squared error

Figure 4.1 compares the transformer results on the seconds and minutes dataset with the LSTM model. The error for the transformer model is less than the LSTM model. The second dataset's error is less than the minute dataset's because the larger size of the second dataset model can train better. Table 4.1 is used to make Figure 4.1.

4.2 Mean squared errors for different number of layers

Comparison by number of Layers					
PWS	Layers = 1	Layers = 2	Layers = 3	Layers = 4	Layers = 5
1	3.3	68.58	68.72	68.55	68.62
5	3.62	68.57	68.71	68.54	68.62
10	3.21	68.04	68.18	68.01	68.09
20	2.87	66.34	66.47	66.32	66.39
30	2.77	65.73	65.86	65.71	65.78
60	2.62	63.68	63.78	63.66	63.71

Table 4.2: Comparing Mean squared error of Transformer models at different number of layers (results are x$10^{-}3$)

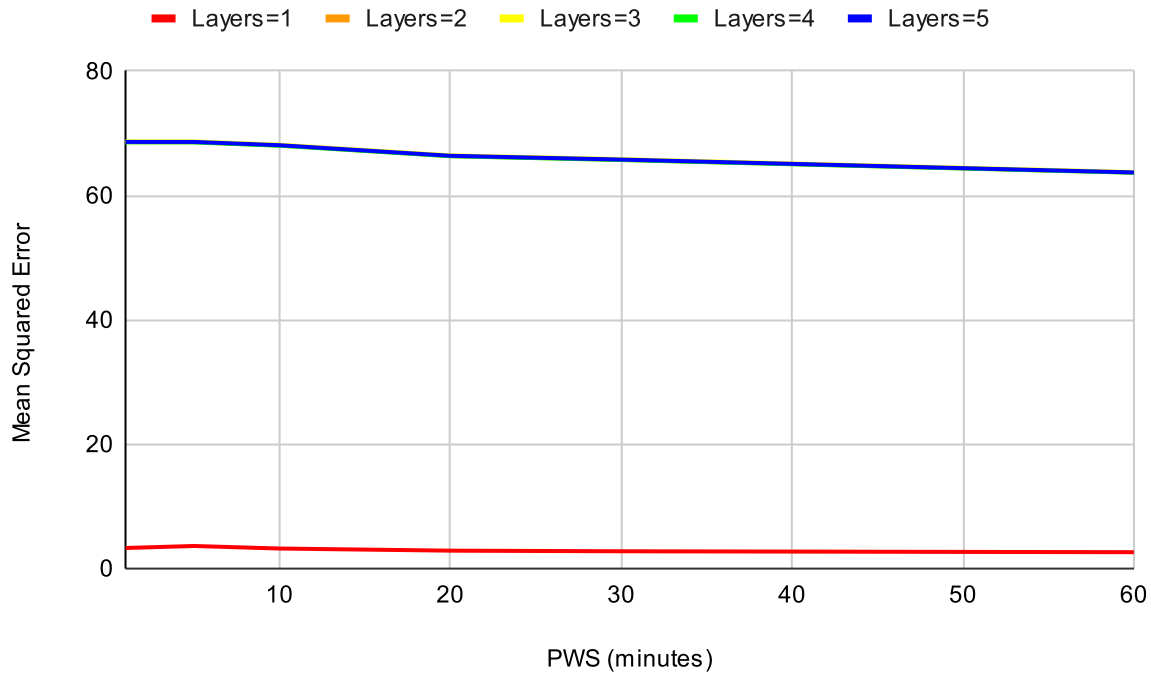


Figure 4.2: Mean squared error by number of layers

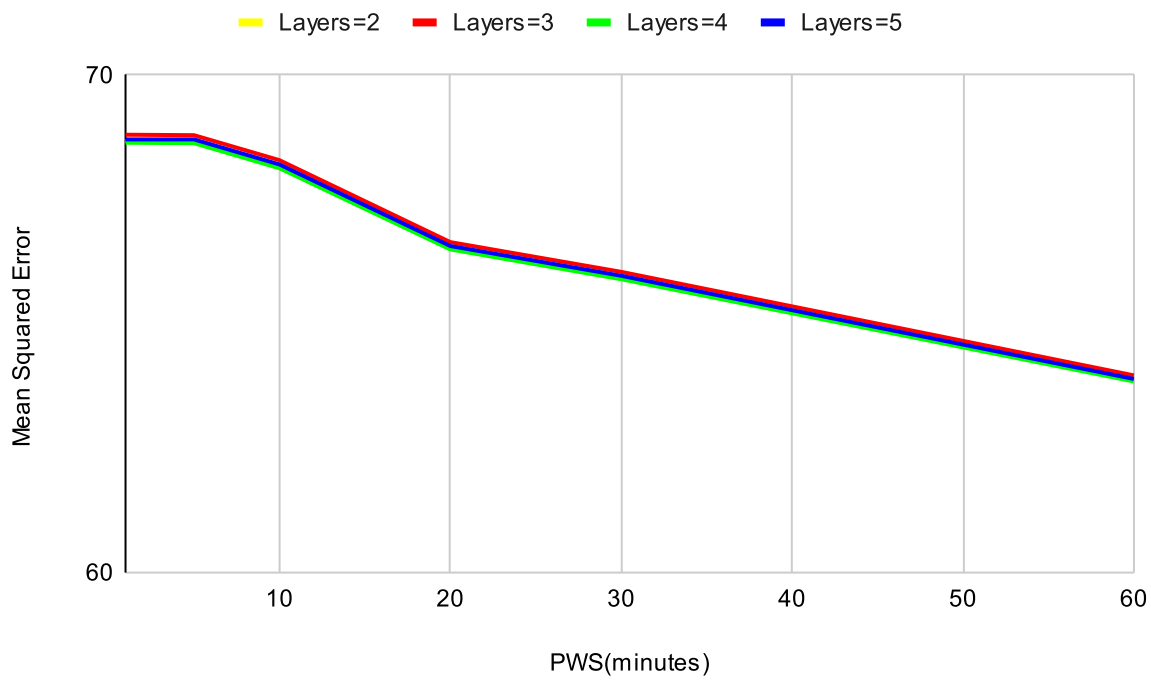


Figure 4.3: Mean squared error by number of layers

Figure 4.2 and 4.3 is the result of the comparison between the mean squared error and the number of layers. Due to the very high difference in error between layer one and other layers, error graphs for layers 2, 3, 4 and 5 are merged into one. Therefore plotted a separate graph comparing the error of layers 2, 3, 4 and 5. We can see that error increases with the number of layers. This is due to an increase in vanishing gradient problems and the inability to larger train models on a smaller dataset.

Chapter 5

Conclusions

We conclude that the transformers are better than LSTMs in the case of time-series prediction. Based on our empirical results, we found that the transformer gives lower prediction error, lower training time, and lower prediction time. We can increase the efficiency and energy consumption of the cloud by using a transformer model.

References

Oliver Guhr. Github: Transformer time-series prediction, 2021. URL `https://bit.ly/3M2mZGs`.

Natasha Klingenbrunn. Medium: Transformers for time-series forecasting, 2021a. URL `https://bit.ly/3yo1uMq`.

Natasha Klingenbrunn. Github: Transformers time-series forecasting, 2021b. URL `https://bit.ly/39dEB3P`.

Jitendra Kumara, Rimsha Goomerb, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *6th International Conference on Smart Computing and Communications ICSCC 2017, 7-8 December 2017, Kurukshetra, India*, 7: 1–7, 2017.

Lennart Svensson. Chalmers university: Self-attention - an introduction, 2020. URL `https://bit.ly/39TchnC`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.*, 15:1–15, 2017.