Energy Time-Series Forecasting

A study of energy consumption prediction using Long Short-Term Memory Neural Networks

December 23, 2021

Tejas Sharma

ts3405

EAEE4000 Final Project Report

Contents

Introduction

Energy consumption is a volatile quantity. It can vary from day to day. It is difficult to plot the trends of the energy consumption on a graph and sudden spikes or deviations from the regular are unavoidable in the daily routine of consumption.

Electricity generation companies have a mammoth task on their hands to deliver uninterrupted power to the entire country with being able to handle uncertainties like outages and generator failures, amongst other such mishaps. If the number of variables, that can be troublesome, can be reduced it proves to be of huge benefit for these companies. Hence, these companies spend a lot of resources in predicting and analyzing such variables, so that they can prepare for what's to come.

One such variable is the energy consumption for the region for the upcoming time periods. This is of vital importance because, this can help companies prepare their resources for the load that is about to be put on the grid and prepare ample amounts of energy for the consumers to continue on with their daily routines.

This project aims to work on a simple univariate problem of energy consumption, that can if handled correctly and calculated efficiently, predict the future energy consumption for the region in concern.

Dataset

The dataset that is used for this project is the AEP hourly energy consumption dataset. American Electric Power (AEP), is a major investor-owned electric utility in the United States, delivering electricity to more than five million customers in 11 states. The dataset is a collection of energy consumptions (MW) for the region, over the period of 121,269 datapoints, ranging from October 2004 to August 2018. The dataset has 2 columns, one for the record of date and time of the data, second for the actual consumption value.

	Datetime	AEP_MW
0	2004-10-01 01:00:00	12379.0
1	2004-10-01 02:00:00	11935.0
2	2004-10-01 03:00:00	11692.0
3	2004-10-01 04:00:00	11597.0
4	2004-10-01 05:00:00	11681.0

Figure 1: AEP hourly dataset head

	AEP_MW
count	121269.000000
mean	15499.651721
std	2591.323421
min	9581.000000
25%	13630.000000
50%	15310.000000
75%	17200.000000
max	25695.000000

Figure 2: AEP hourly dataset description

```
In [19]: plt.plot(d['Datetime'], d['AEP_MW'])
Out[19]: [<matplotlib.lines.Line2D at 0x21604430040>]
```

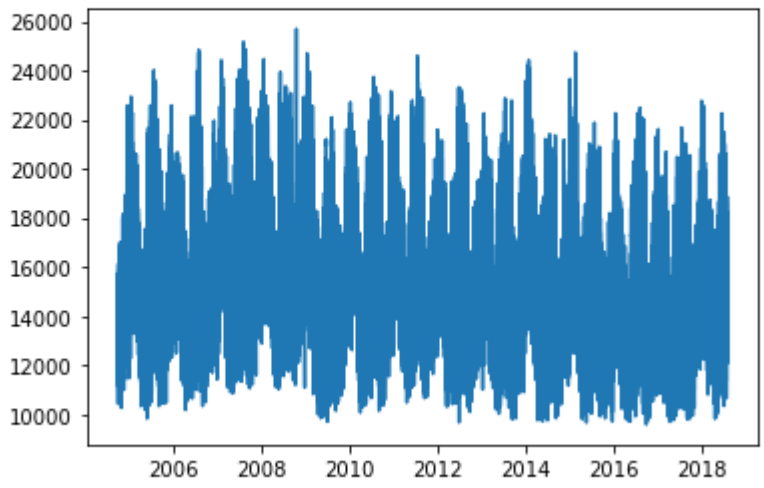

Figure 3: AEP hourly consumption plot

Methodology

The methodology that will be followed for the prediction of this value will be Long Short-Term Memory (LSTM) architecture. This is a type of Recurrent Neural Network (RNN) (Sherstinsky, 2018) which helps predict time-series easily. Since, the dataset selected records the variables against time, each hour of the 14 years in question, this dataset can be handled by LSTMs and future predictions can be made.

The dataset at hand will be split into training and validation sets, in order to observe the accuracy of the model in construction. This means, the dataset would be split into 2 parts, one of which would be introduced to the model to train itself on, and make predictions of the next series, which is the validation set. This validation set is hidden from the model and would not be disclosed until the model has made its prediction, only to superimpose the predictions on the validation set and determine the accuracy. Then, this model would be applied to predict future values which are previously unknown.

Given the time constraints and the computational power constraints, this model would not be as effective in real world scenarios as it would still be considerably inaccurate. Also, real world scenarios would have more than 1 variables on which the energy consumption would depend on. Hence, the basic model construction would be useless for that scenario.

Architecture

Recurrent Neural Networks

A recurrent neural network (RNN) (IBM Cloud Education, 2020) is a kind of artificial neural network which utilizes sequential or time-series data. Such deep learning algorithms are usually used for ordinal or temporal problems, which may include language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Bixby, voice search, and Google Translate. Like feed-forward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

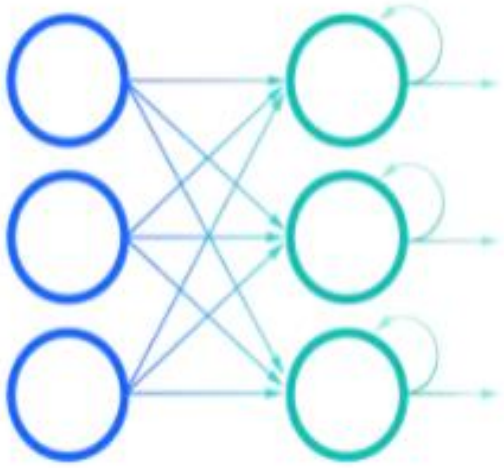

Figure 4: RNN Model

Long Short-Term Memory

LSTM (Sepp Hochreiter, 1997) is a popular RNN architecture, which was introduced as a solution to vanishing gradient problem. In this, the architecture attempts to address the problem of long-term dependencies. That is, if the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. To remedy this, LSTMs have “cells” in the hidden layers of the neural network, which have three gates–an input gate, an output gate, and a forget gate. These gates control the flow of information which is needed to predict the output in the network.

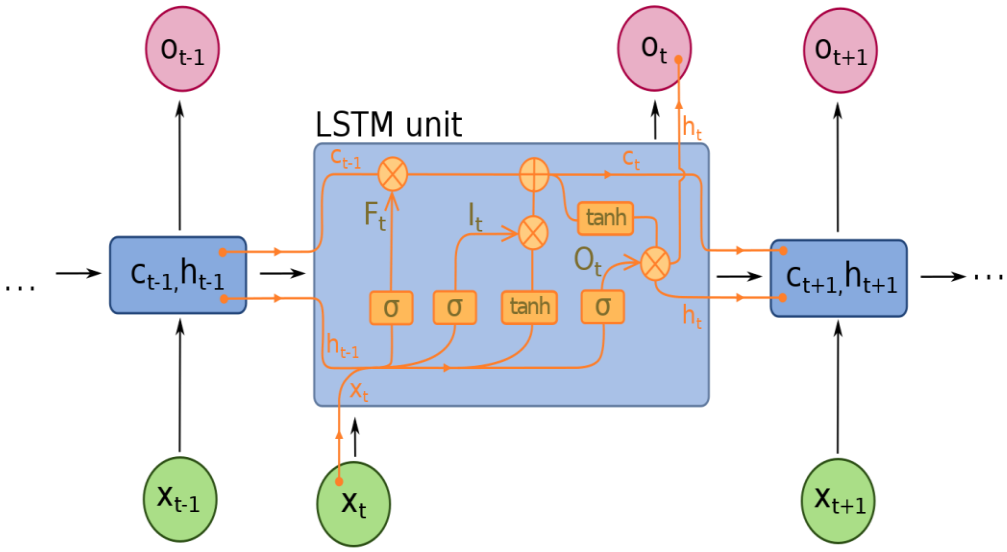

Figure 5: LSTM Model

This project would aim to apply LSTM architecture to study the time-series of energy consumption in the regions that are supplied energy from AEP. The dataset, having time-series data would serve as the input for the architecture which will then result in the future energy consumption predictions as the output.

Terminology

LSTM layer depth – number of LSTM hidden layers stacked over each other in one Neural Network

Dense – number of neural networks that feed their output to the next neural network

Lag – hours in the past, which is visible to the model

Batch-size – the number of training examples used in one iteration

Model Overview

In this study the LSTM model was implemented with the help of keras API. Using trial and error, the parameters were changed to observe the changes in prediction and loss values.

The parameters that were selected for this study were:

1. LSTM layer depth: 64
2. Lag: 48
3. Train - Test split: 90:10
4. Epochs: 10/40
5. Batch size: 256/1024
6. Dense Layers: 128

These parameters were varied over time to observe the changes in the loss that those had.

The code to this model was inspired by the time-series forecasting code by Eligijus Bujokas.

The training and test set were split only when we had to evaluate the viability of the neural network. Then the split was brought down to zero, in order for the neural network to predict future data based on the previous data it was provided.

The activation function used for the neural network was Rectified Linear Unit (ReLU) (Bing Xu, 2015). The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. Hence, it can be written as,

$$f(x) = max(0, x)$$

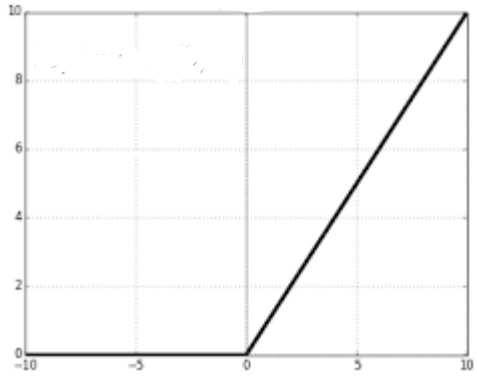

Figure 6: ReLU Activation Function

The optimizer used is the Adam Optimizer (Diederik P. Kingma, 2014). Adam is an adaptive learning rate optimization. The algorithm leverages the power of adaptive learning rates methods to find individual learning rates for each parameter.

The loss function used is “mean squared error” (MSE). It is the difference between the model’s predictions and the ground truth, squared and averaged out across the whole dataset.

It can be shown as,

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

Here, N is the number of samples, y_i is the model’s prediction, and $\hat{y}_i$ is the ground truth.

Results

1. Batch size: 256 -

   The results with different lags and LSTM layer depths gave different predictions for the future energy consumptions. All predictions have been made with 128 dense layers and 10 epochs.

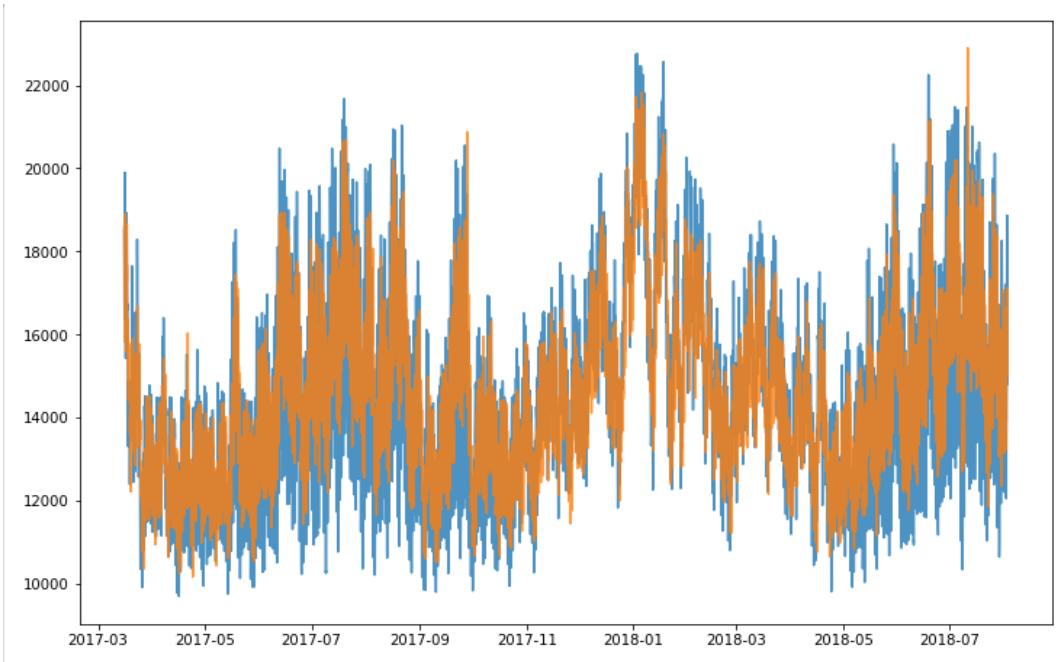

Figure 7: Test set validation; Orange- prediction. Blue- Ground Truth. Lag = 48

The prediction for the future can't be evaluated with any ground truth. Hence, it is tried for with different lags, i.e., the amount of data from the past, in hours, that is known to the model.

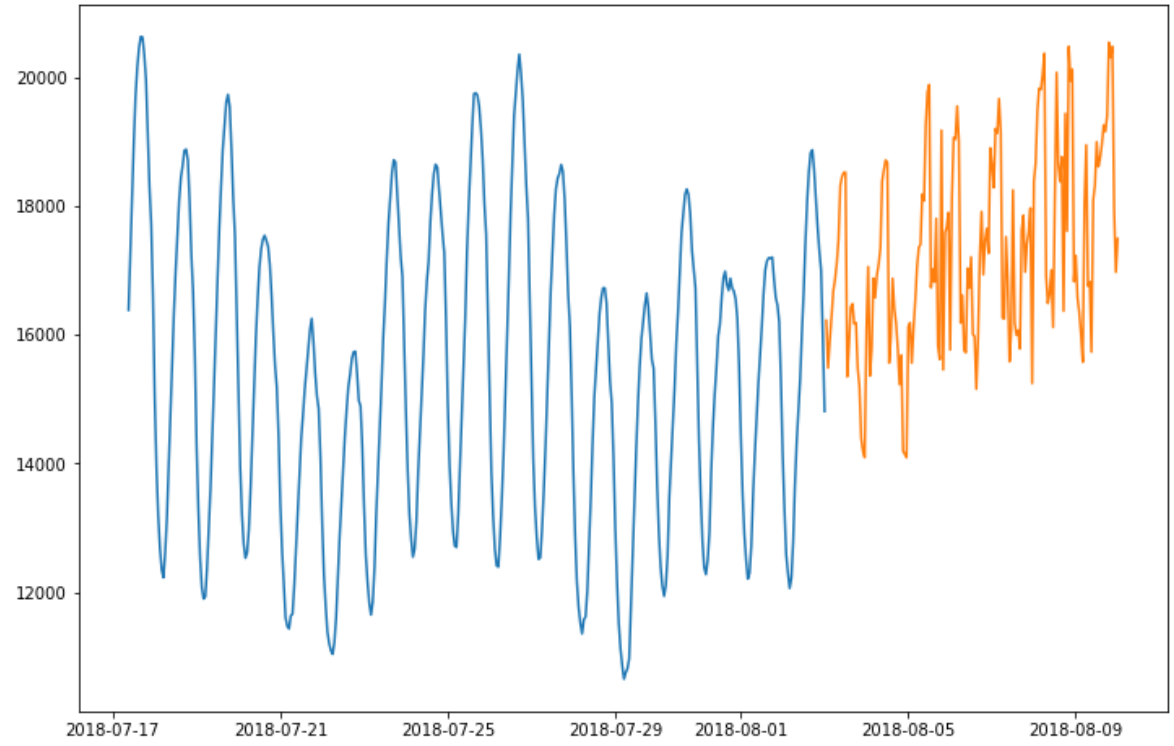

Figure 8: Forecast, lag = 72

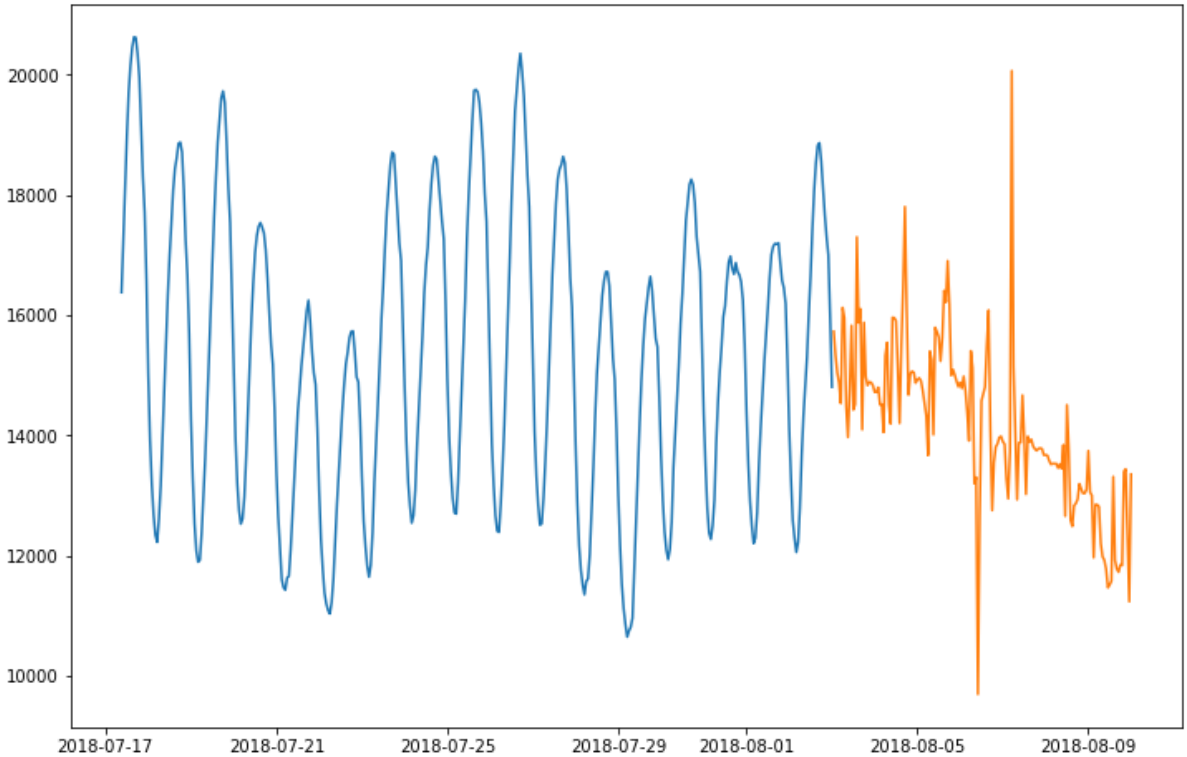

Figure 9: Forecast, lag = 96

2. Batch size: 1024

   Increasing the batch size gave better validation results. Best results were observed when, batch size was 1024 (Higher batch sizes could not be tested due to computational limitations). These values were tested over 40 epochs to give the model enough tries to correct its loss. The loss was reduced from around 138 million to now about 196 thousand over the course of 40 epochs.

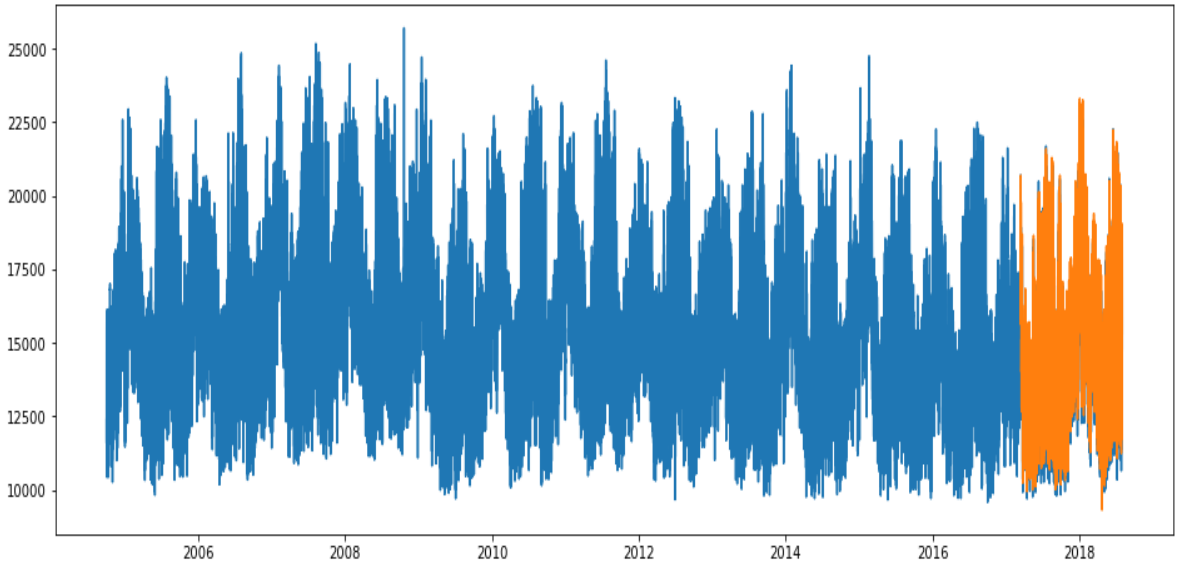

Figure 10: AEP dataset with prediction in orange.

```
Epoch 40/40
427/427 [==============================] - 34s 80ms/step - loss: 196209.5469 - val_loss: 165574.7969
```

Figure 11: loss and val_loss values for model fitting

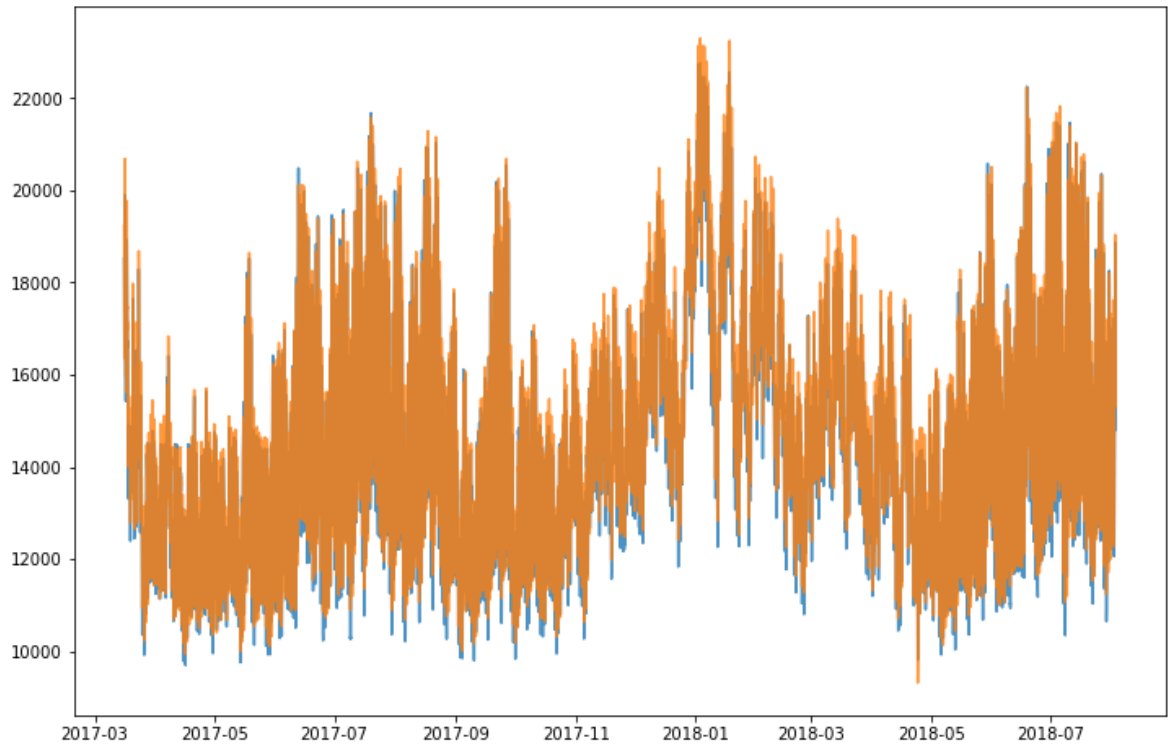

Figure 12: test set validation; Orange: Prediction, Blue: Ground Truth. Lag = 48

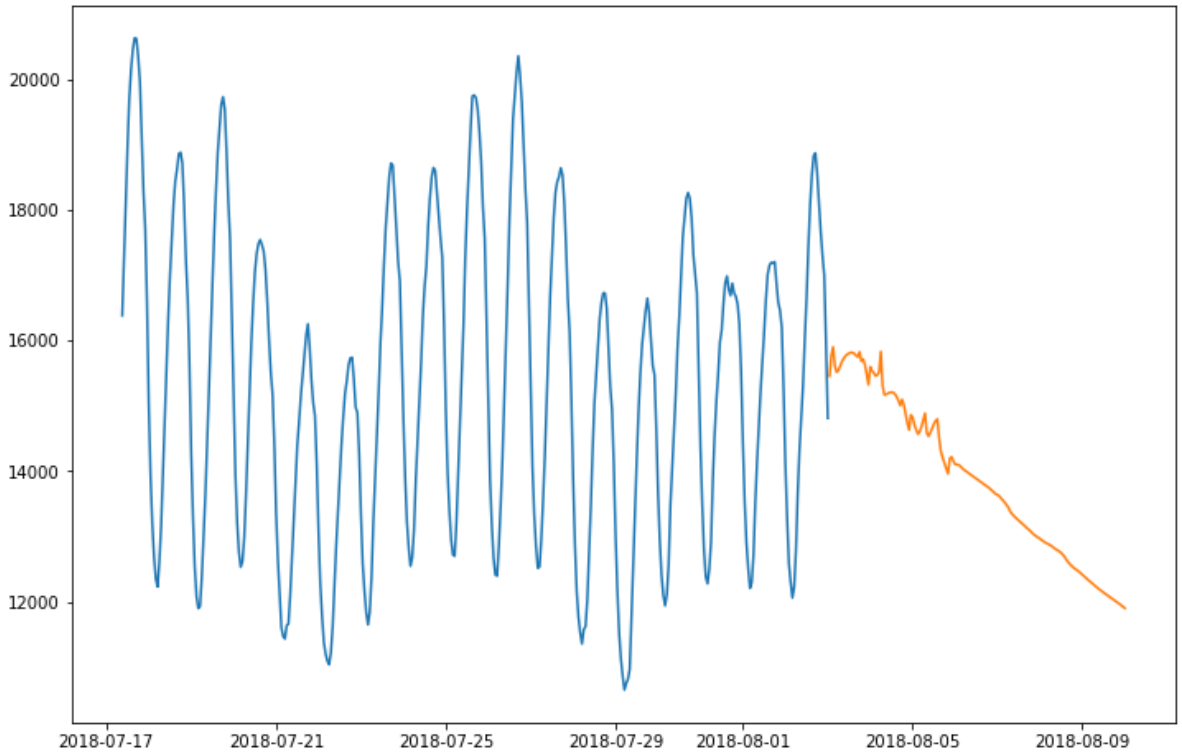

Figure 13: Forecast over 40 epochs. lag = 72

```
Epoch 40/40
474/474 [==============================] - 53s 111ms/step - loss: 2529641.2500
```

Figure 14: loss value for forecast over 40 epochs

Discussion

As can be seen from the above results. The increase in batch size and the increase in epochs give better validation results from the model. The forecast however, cannot be accurately figured out by parameter tuning, as each tuning gave a result that could not be extrapolated into a function of improvement. But increasing the amount of lag, or the data that is visible to the model for forecasting has resulting in getting a more controlled result, as seen in figure 8 and 9. Lower lag results have the tendency to deviate into uncertain regions.

In future studies, attempt can be made to study the deviation of the forecasts into the uncertain regions. Dropout can be implemented to try to curb this effect. Also, instead of LSTMs, an alternate architecture can be used, known as the Transformer (Ashish Vaswani, 2017). Transformer is strong for applications like NLP. In order to implement Transformers to time-series applications, a Time2Vec (Seyed Mehran Kazemi, 2019) modification has to be applied which helps it take the time input in the place of its positional encoders.

References

Ashish Vaswani, N. S. (2017). Attention Is All You Need.

Bing Xu, N. W. (2015). *Empirical Evaluation of Rectified Activations in Convolution.*

Diederik P. Kingma, J. B. (2014). *Adam: A Method for Stochastic Optimization.*

IBM Cloud Education. (2020). *Recurrent Neural Networks.*

Sepp Hochreiter, J. S. (1997). *LONG SHORT-TERM MEMORY.*

Seyed Mehran Kazemi, R. G. (2019). Time2Vec: Learning a Vector Representation of Time.

Sherstinsky, A. (2018). *Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.*