# Prediction using Traffic Flow data

## Introduction

Purpose of this assignment is to analyse a Traffic Flow dataset using a variety of Deep Learning techniques. A model will be trained capable of predicting the carriageway flow of a road given a certain date and time.

Information of this sort can be utilized both from individual travellers to private companies or even state authorities. The value of Traffic Flow prediction can be acknowledged easily if one were to think of the support this information could give to commuters. They would be able to create an improved travel plan avoiding traffic, reducing carbon emissions and fuel consumption.

On a larger scale the benefits can also be seen with the de-congestion of main traffic roads and a general improvement in traffic circulation.

## Indicative Research Publications

- **Traffic Flow Prediction With Big Data: A Deep Learning Approach** by Yinsheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li and Fei-Yue Wang
- **Deep learning for short-term traffic flow prediction** by Nicholas G. Polson, Vadim Sokolov

## Task Definition

A regression model has been implemented that predicts the Average Carriageway Flow of a certain road given both continuous and categorical values. The dataset used to train this model is comprised of a years Traffic Flow Data generated monthly from a certain highway in England and saved in a .csv file.

Apart from identifiers and descriptions of each site, each file contains information about the Day Type (whether it's a working day or a holiday), the Average Speed, the Average Flow detected on any lane and a Quality index of the data provided. The data provided are generated with 15 minute intervals.

| | Local Date | Local Time | Day Type ID | Total Carriageway Flow | Total Flow vehicles less than 5.2m | Total Flow vehicles 5.21m - 6.6m | Total Flow vehicles 6.61m - 11.6m | Total Flow vehicles above 11.6m | Speed Value | Quality Index | Network Link Id | NTIS Model Version\n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2018-04-01 | 00:14:00 | 5 | 65.0 | 47.0 | 1.0 | 3.0 | 14.0 | 101.25 | 15 | 126001501 | 7 |
| **1** | 2018-04-01 | 00:29:00 | 5 | 71.0 | 59.0 | 3.0 | 0.0 | 9.0 | 99.63 | 15 | 126001501 | 7 |
| **2** | 2018-04-01 | 00:44:00 | 5 | 55.0 | 40.0 | 4.0 | 3.0 | 8.0 | 100.40 | 15 | 126001501 | 7 |
| **3** | 2018-04-01 | 00:59:00 | 5 | 66.0 | 49.0 | 6.0 | 4.0 | 7.0 | 100.41 | 15 | 126001501 | 7 |
| **4** | 2018-04-01 | 01:14:00 | 6 | 55.0 | 34.0 | 5.0 | 3.0 | 13.0 | 98.90 | 15 | 126001501 | 7 |

Image 1: Preview of the Dataset imported

# Description

Two distinct models have been created depending on their short-term and long-term ability to predict.

First the *Long-Term Model* (LTM) is trained. Goal of this model is to be able to make predictions for any given date and time. The preprocessing function *ltm_proc* handles the training features. The input features are **Day Type** (a categorical integer describing the type of day) and **Local Time** (the local time).

Because **Day Type** contains categorical values, an Entity Embedding approach has been followed, mapping the distinct variables into an n-dimensional vector. This method was attempted because of its potential to limit the number of columns needed per category and to meaningfully represent categories in the transformed space.

For the 2nd feature, **Local Time**, acknowledging it as a cyclical feature, it is important to find a way to represent the values in a manner that preserves the connection between them. For this reason, through the *time_handler* function this feature has been replaced by two continuous features **Sine** and **Cosine Weight** representing the cyclical feature in the form of coordinates on a circle.

| | Day Type ID | Sine Weight | Cosine Weight |
|---|---|---|---|
| **0** | 5.0 | 0.000000 | 1.000000e+00 |
| **1** | 5.0 | 0.065403 | 9.978589e-01 |
| **2** | 5.0 | 0.130526 | 9.914449e-01 |
| **3** | 5.0 | 0.195090 | 9.807853e-01 |
| **4** | 6.0 | 0.258819 | 9.659258e-01 |

Image 2: Preview of the training data using the first preprocessing function

The 2nd model trained is the *Short-Term Model* (STM), purpose of which is to make live predictions about the current traffic flow using previously collected data. Function *stm_proc* handles the training features of this model. This approach utilizes information chosen with, a user defined, number of *k* (integer, k>2) previous 15min steps.

Apart from the previous two features (**Day Type** and **Local Time**), the *stm_preproc* function three more continuous features have been added. **Previous Step** feature computes the quantity and the direction of the Average Traffic Flow in last *k* quarters, in other words it is the difference between the previous Average Traffic Flow (ATF) and the *i-k* ATF where i is the current position. **Previous Flow** and **Previous Speed** are the other two features containing the the immediately preceding ATF and Average Speed Value respectively.

Note that, in both functions, rows with NaN values have been dropped and all continuous values have been standardized with the function *standardization*.

| | Day Type ID | Sine Weight | Cosine Weight | Previous Step | Previous Flow | Previous Speed |
|---|---|---|---|---|---|---|
| **0** | 5 | 0 | 1 | -0.000590692 | -1.36446 | 0.488457 |
| **1** | 5 | 0.0654031 | 0.997859 | -0.000590692 | -1.36446 | 0.488457 |
| **2** | 5 | 0.130526 | 0.991445 | 0.10257 | -1.34157 | 0.339237 |
| **3** | 5 | 0.19509 | 0.980785 | -0.275686 | -1.40262 | 0.410163 |
| **4** | 6 | 0.258819 | 0.965926 | 0.188538 | -1.36065 | 0.411084 |

Image 3: Preview of the training data using the second preprocessing function

Finally, both LTM and STM follow the same methodology to build the final model. In function *embed_model* two models are created which are later concatenated into the main model. This happens in order to combine the Embedded categorical data with the continuous data (more information about Entity Embeddings is provided in the sources).

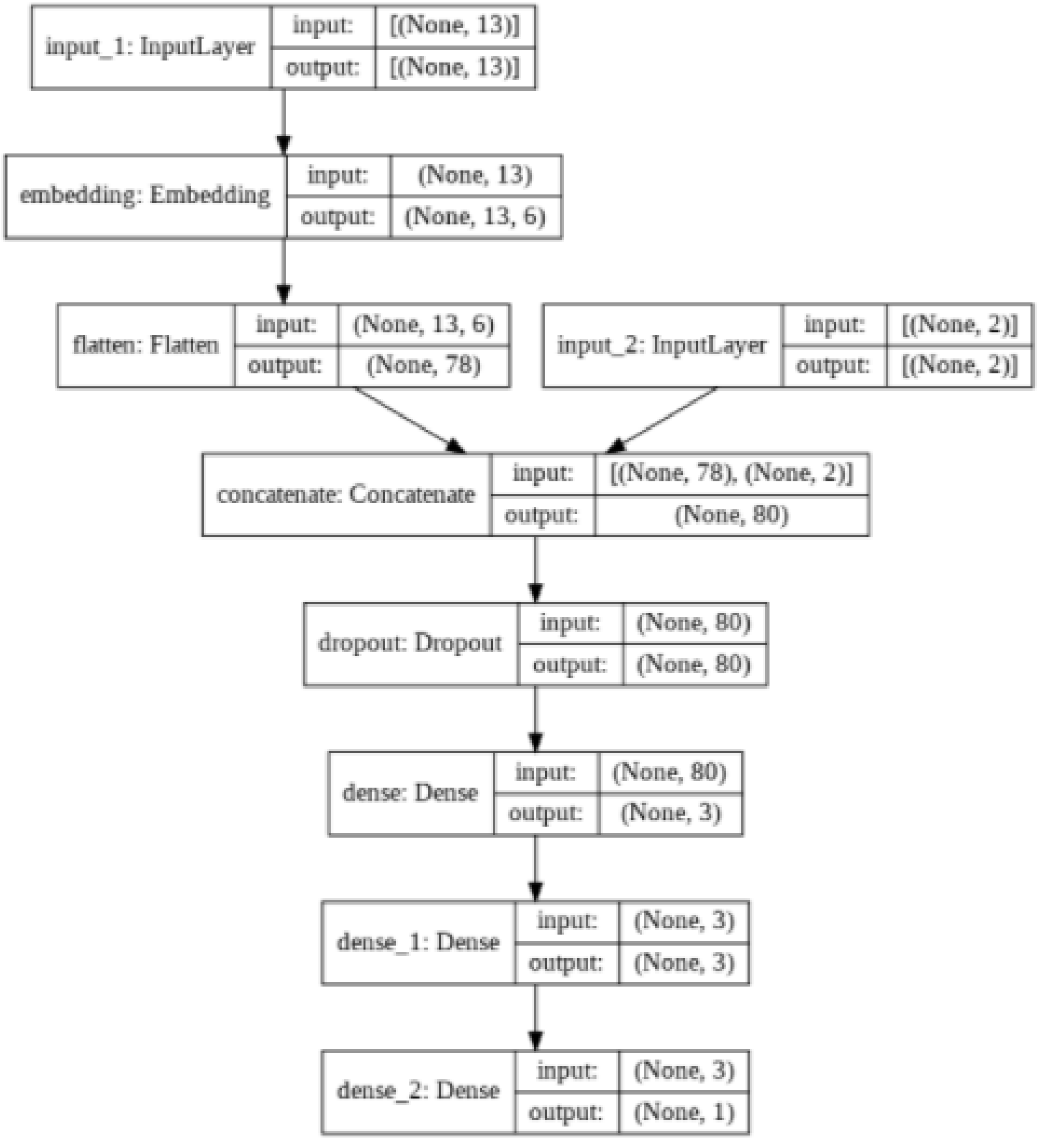



Image 5: A plot summary of the main models' structure

# Experimental Evaluation

For the evaluation of the regression model **Mean Absolute Error** and **Root Mean Squared Error** have been employed . Following is a list of experiments with different hyperparameters, based on the model of Image 5, and their results.

For Experiments 1-5 Train/Test size is 50/50 and k=2.

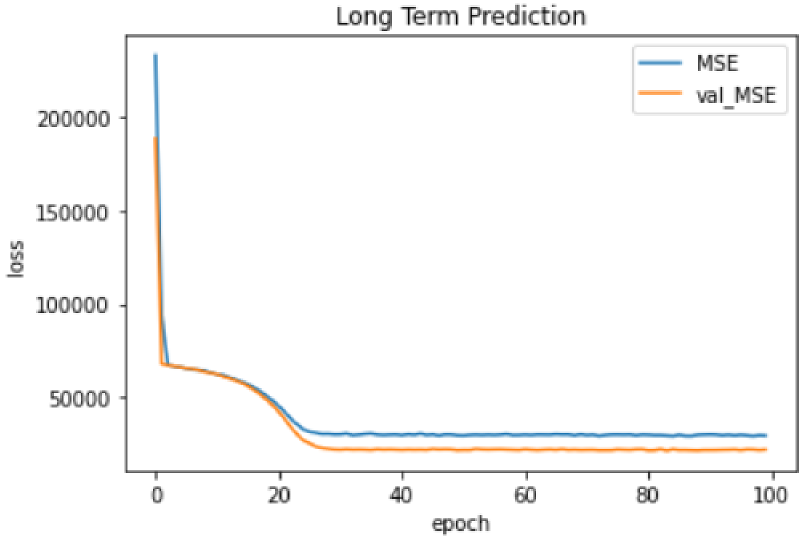
## Experiment 1

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *RELU* | *normal* | *3, 3** | *Adam* | *32* |

*two layers of 3 neurons each

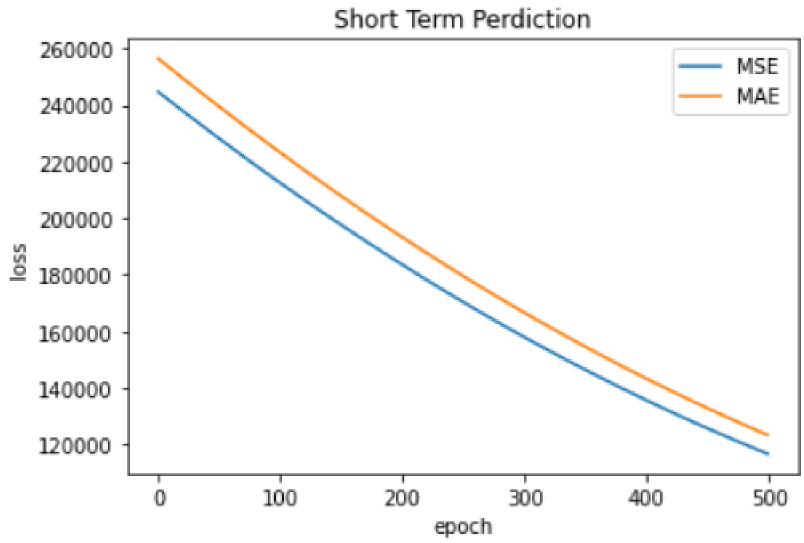
## Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *20707* | *117* | *144* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *98574* | *255* | *313* |




Observations:

This scenario points out that there is much room for improvement. The LTM manages to complete its training in under 30 epochs with MAE=117 while the STM has still not completely training after 500 epochs with MAE=255.

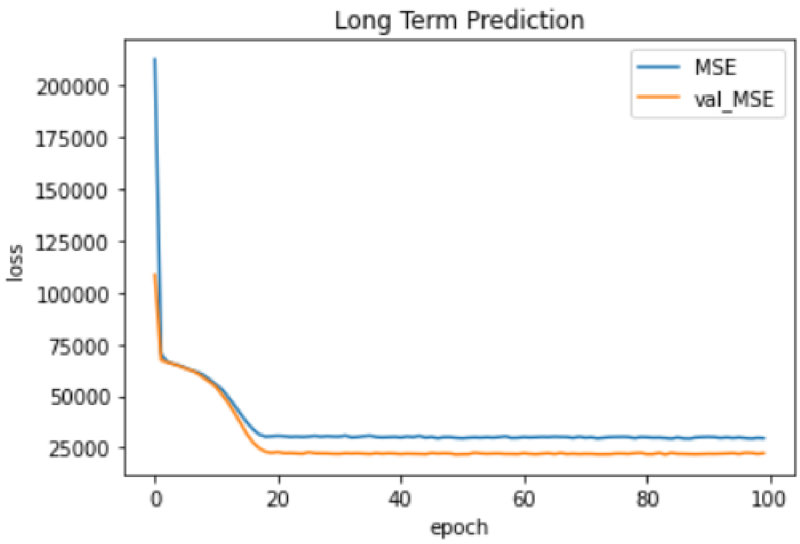
# Experiment 2

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *RELU* | *normal* | *3, 300** | *Adam* | *256* |

*two layers of 3 and 300 neurons respectively

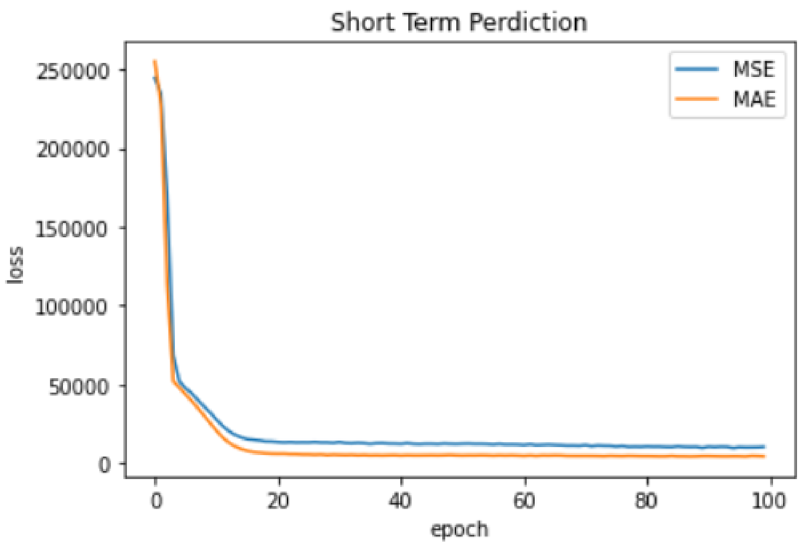
## Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *20323* | *116* | *142* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *4851* | *53* | *69* |



Observations:
In this scenario both models managed to complete their training close to 20 epochs. By increasing batch size and neurons inside one of the dense layers both models managed to train faster. On the other hand, there was no significant improvement for the LTM upon evaluation when the STM, having been fully trained, gave better results.

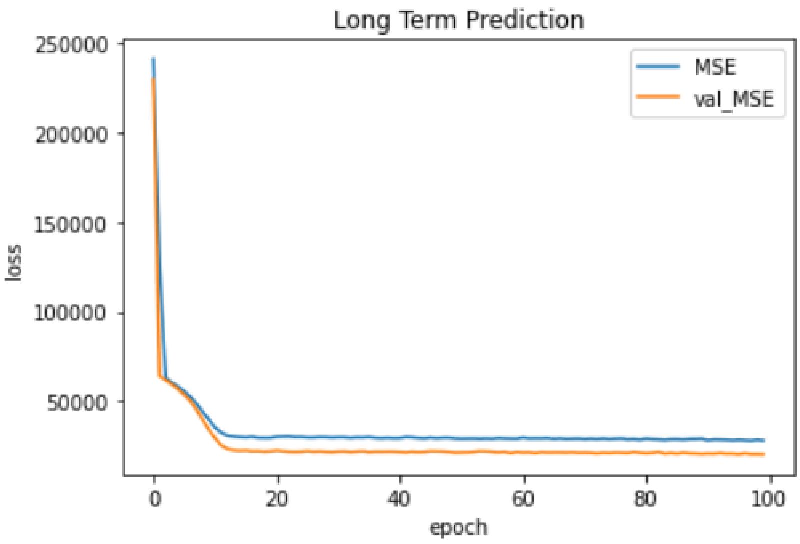
# Experiment 3

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *RELU* | *normal* | *3, 30** | *Adam* | *256* |

*two layers of 3 and 30 neurons respectively

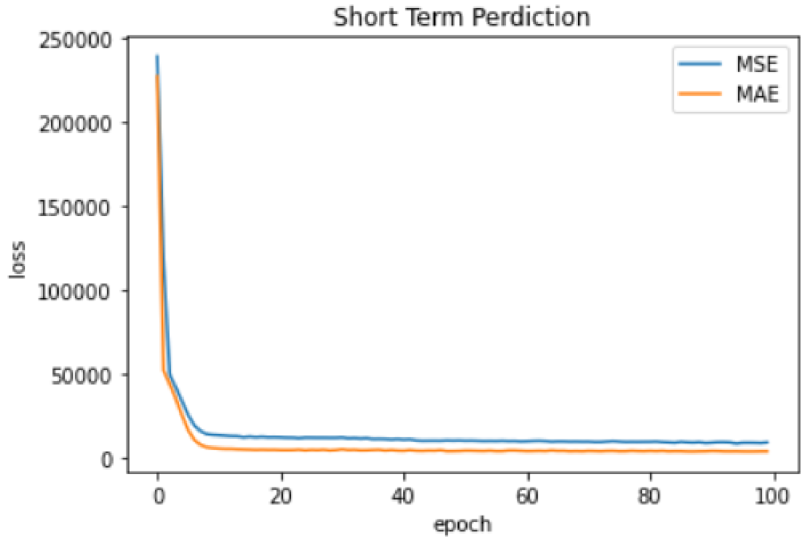
## Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *20923* | *118* | *145* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *4834* | *53* | *70* |




<u>Observations:</u>
In this case whilst there is a change in the number of neurons of the second layer, from 300 to 30, there is no substantial difference in the evaluation metrics in either model.

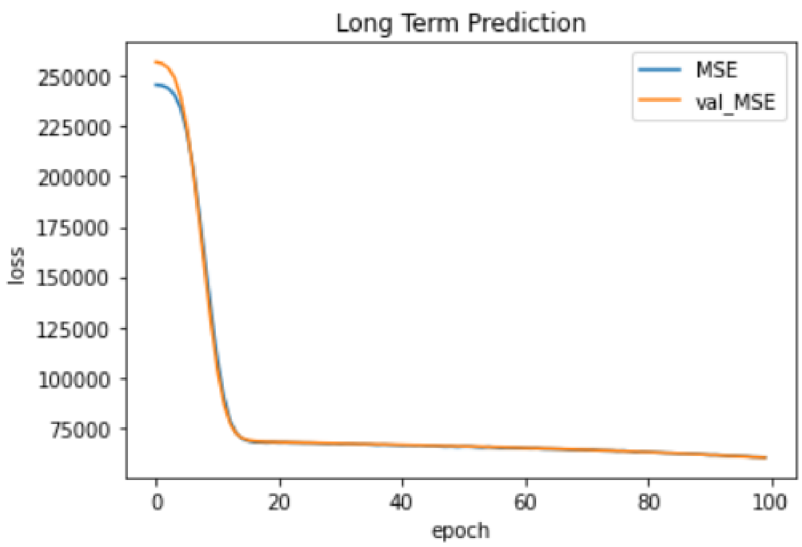
# <u>Experiment 4</u>

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *RELU* | *normal* | *3** | *Adam* | *256* |

*only one layer

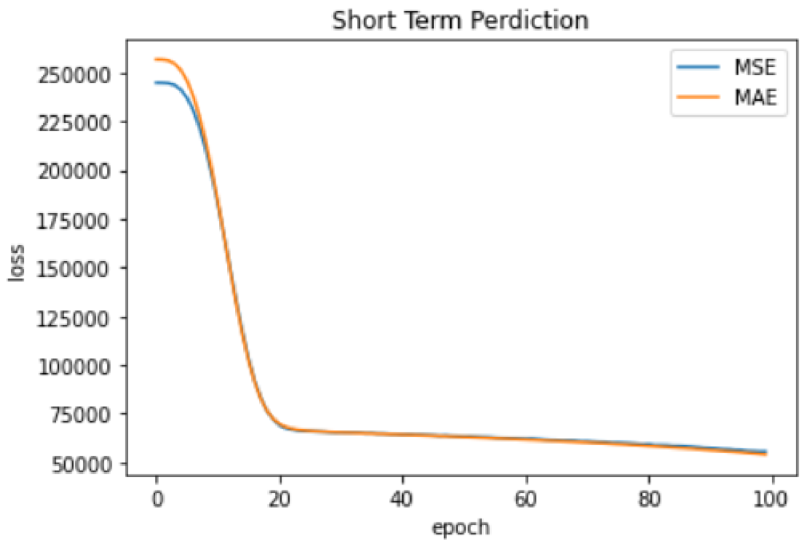
## <u>Long-Term Model Evaluation:</u>

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *54526* | *210* | *234* |



## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *49209* | *200* | *222* |




**Observations:**

In this experiment one Dense layer was removed. It is clear that the models are not as efficient.

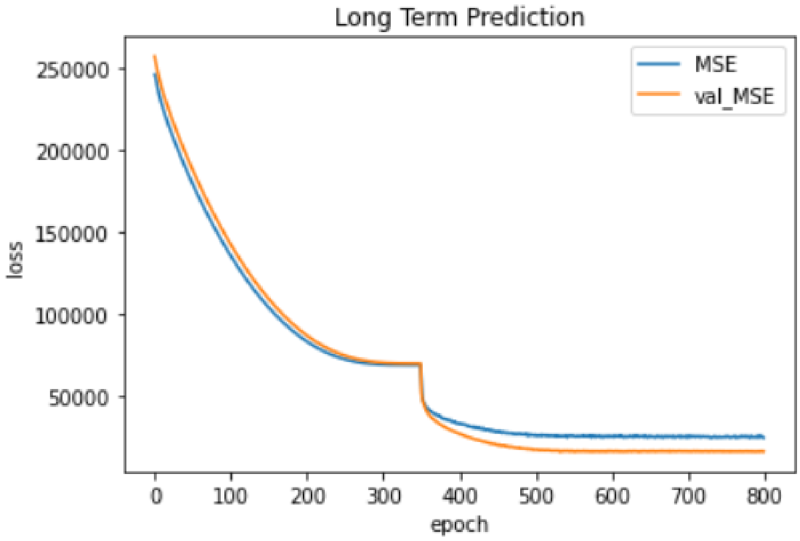
# Experiment 5

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *tanh* | *normal* | *3, 30** | *Adam* | *256* |

*two layers of 3 and 30 neurons respectively

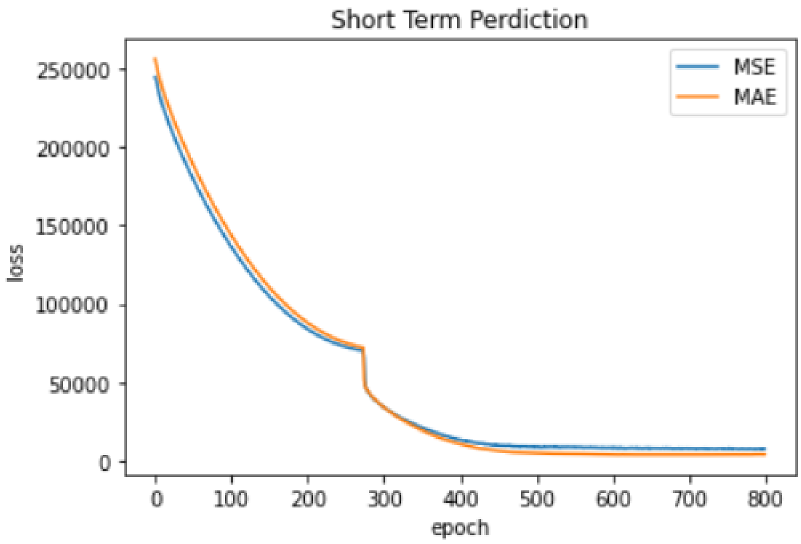
## Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *16810* | *98* | *129* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *5367* | *57* | *73* |




## Observations:

For this experiment *tanh* was used as activation function. The STM has very similar evaluation results compared to *relu* and has even better results for the LTM but requires far more epochs of training.

For Experiment 6 Train/Test size is 80/20 and k=2.

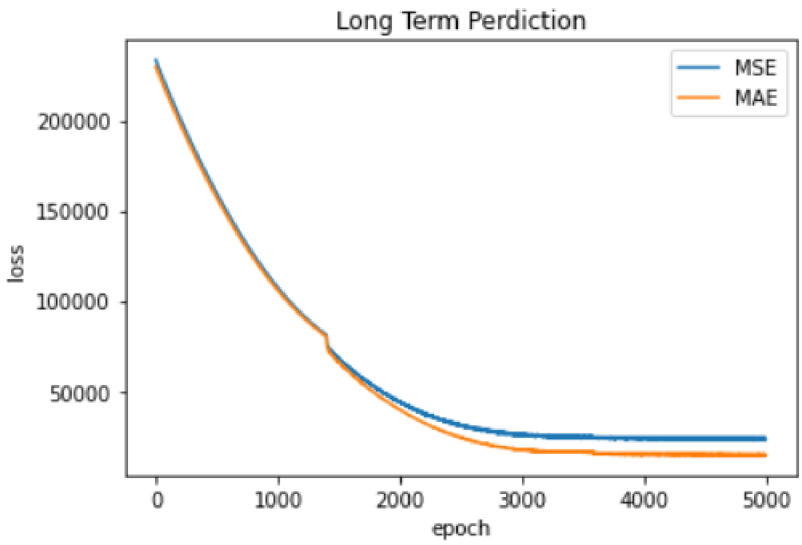
# Experiment 6

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *tanh* | *normal* | *3, 30** | *Adam* | *256* |

*two layers of 3 and 30 neurons respectively

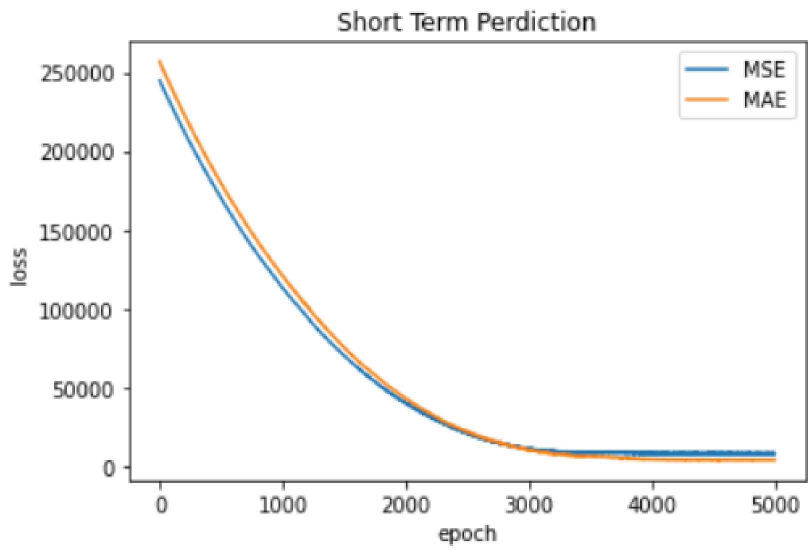
## Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *14615* | *94* | *121* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *5018* | *56* | *71* |




Observations:

For this experiment only the Train/Test size was changed. By increasing the training size, from 50% to 80%, there are slightly better evaluation results but the model took much more time to complete its training.

For Experiment 7 Train/Test size is 50/50 and k=3.

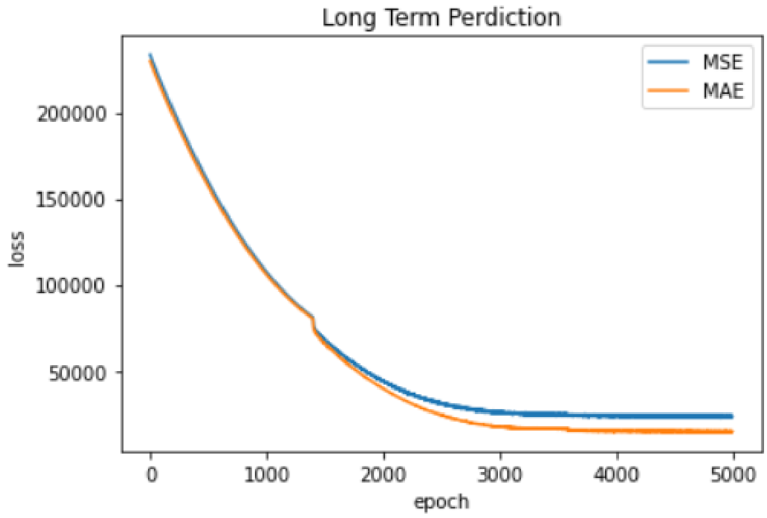
## Experiment 7

| Activation Function | Kernel Initializer | Neurons per Layer | Optimizer | Batch Size |
|---|---|---|---|---|
| *tanh* | *normal* | *3, 30** | *Adam* | *256* |

*two layers of 3 and 30 neurons respectively

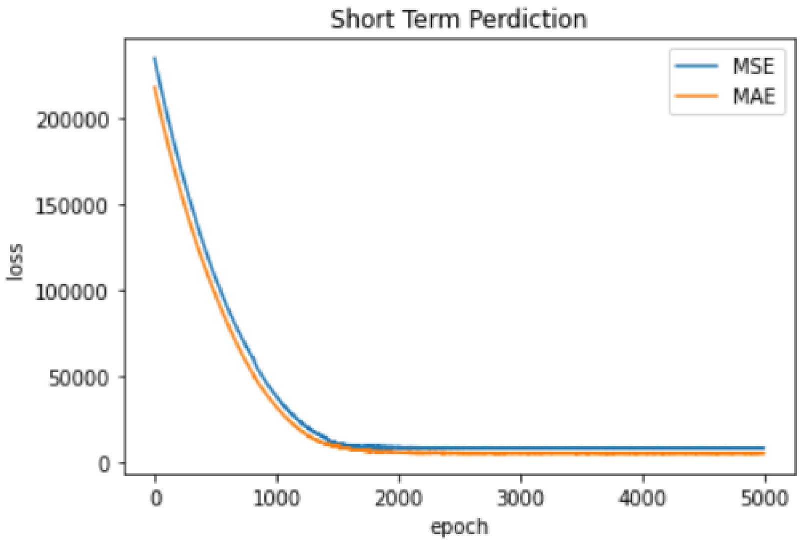
### Long-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *14615* | *94* | *121* |




## Short-Term Model Evaluation:

| Loss (MSE) | MAE | RMSE |
|---|---|---|
| *1203* | *24* | *35* |




Observations:

For this last experiment Train/Test size is 50/50 and k is set to 3. The LTM has the exact same evaluation results with Experiment 6 where the only difference is that Train/Test size is 80/20. On the contrary the STM evaluation results are the best so far, which means that k is a parameter worth examining.

# Conclusions

By analysing the the different hyperparameters used in the previous experiments a few assumptions can be made. The best overall results for both models were observed in Experiment 7. The Short Term Model tended to have better evaluation results than the Long Term Model. This was expected since the LTM had much fewer features to train on. Following is a plot comparing the STM to the LTM.

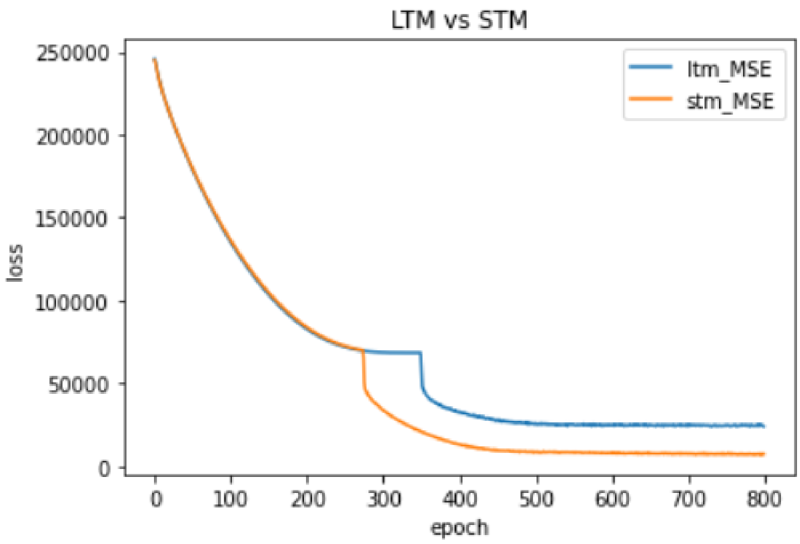



Image 6: Plot of Experiment 5, STM vs LTM

It is worth noting that all experiments are just a sample of the hyperparameter tuning that this model could undergo. The hyperparameters were chosen based on Google Colaboratorys' computing ability and time restrictions. At first an effort was made with GridSearchCV but there were compatibility issues with the input so a manual tuning approach was eventually preferred.

A future task could be to apply further hyperparameter tuning with various technologies to better both models.

# Sources and Links

- https://medium.com/@davidheffernan_99410/an-introduction-to-using-categorical-embeddings-ee686ed7e7f9
- https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526
- https://mmuratarat.github.io/2019-06-12/embeddings-with-numeric-variables-Keras
- https://github.com/entron/entity-embedding-rossmann/tree/39b137968f75ac065d62f4703a3a42263858a169
- http://blog.davidkaleko.com/feature-engineering-cyclical-features.html

- **Data Source: Traffic Flow Data** (A1->(04/2018-03/2019)->MIDAS Site - 10285 at A1 northbound between A639 and A1(M) J40 (126001501) https://data.gov.uk/dataset/9562c512-4a0b-45ee-b6ad-afc0f99b841f/highways-england-network-journey-time-and-traffic-flow-data

- **Source Code:** https://colab.research.google.com/drive/116wCc8xZOb_e8oFhRn1PqUcWBwp5_Jk_

Stratakos Efstathios 21549