NAWAAZ SHARIF

CSC 578

ASSIGNMENT 7

JOURNEY:

The start of the final project was quite interesting, as I did many projects in data science, but this was my first time doing it on time series data.

Before starting, I had my curiosity as to how the visualization will be. It was quite similar.

The dataset that we are using in this project is “Metro_Interstate_reduced.csv” data. The columns that the dataset has are – holiday, temp, rain-1h, snow_1h, clouds_all, weather_main, weather_describe, date_time, traffic_volume. The dataset had no NULL values in it.

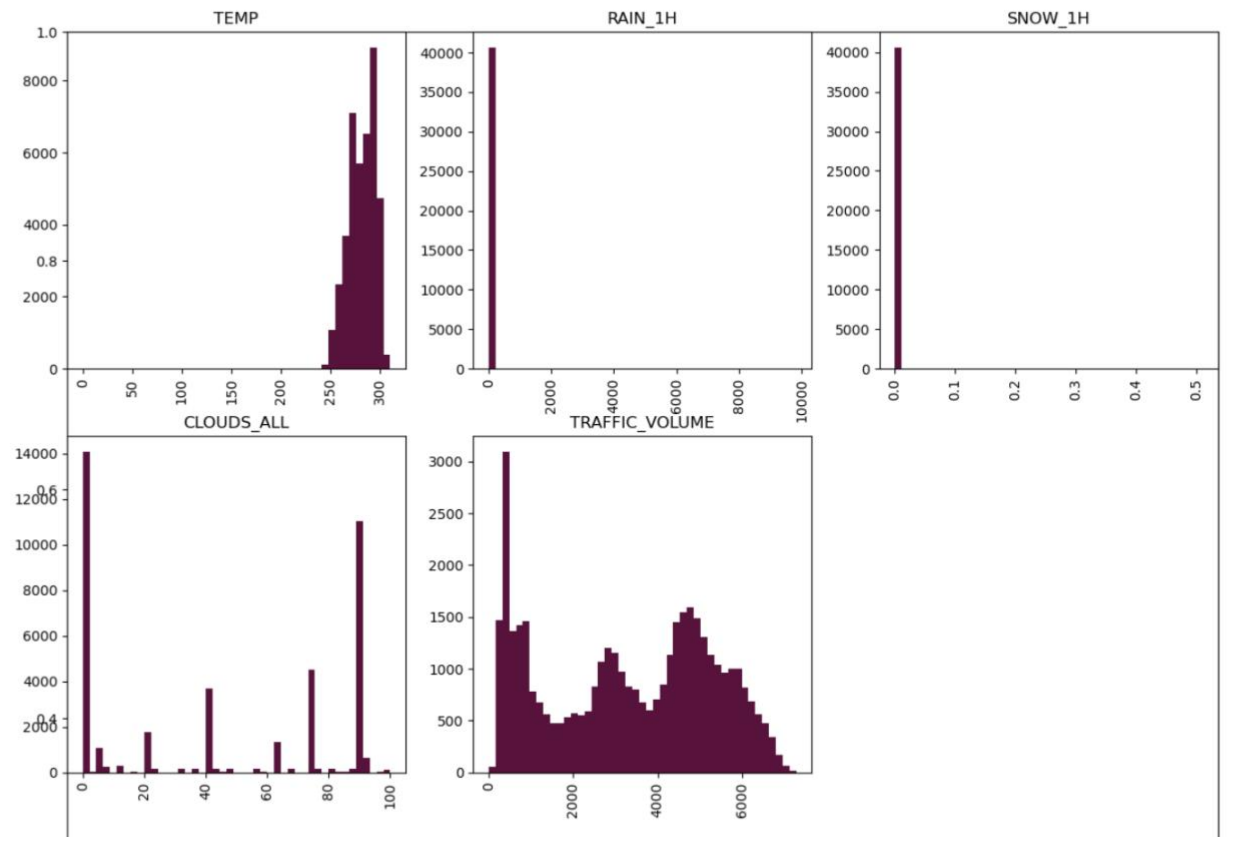
For my first visualization, I took all the numeric attributes of the dataset – temp, rain_1h, snow_1h, clouds_all, traffic_volume and plotted histogram to see the distribution of the data and to see if there is any pattern between the data.

The graph of temp was all on the lest side(higher temp) of the graph, as the values were in kelvin which makes sense for the graph to be like that.

The graph of rain_1h has only single bar on 0 for 40000 mm of rain in 1h, which might be the outlier of our column or a mistake in the entry of the data.

Similarly for the graph of snow_1h which is also kind of same to rain_1h.

The graph of clouds_all shows the distribution of the percent of clouds in the sky, which is distributed as we would expect, unless is sunny or cloudy depending on the season.



I also plotted a correlation graph to check if there's any correlation between the columns; the highest positive correlation was between traffic_volume and temp of 0.14; and the lowest negative value was between traffic_volume and snow_1h.

To proceed with the models, I have dropped the columns – holiday, weather_desciption and weather_main from the dataset.

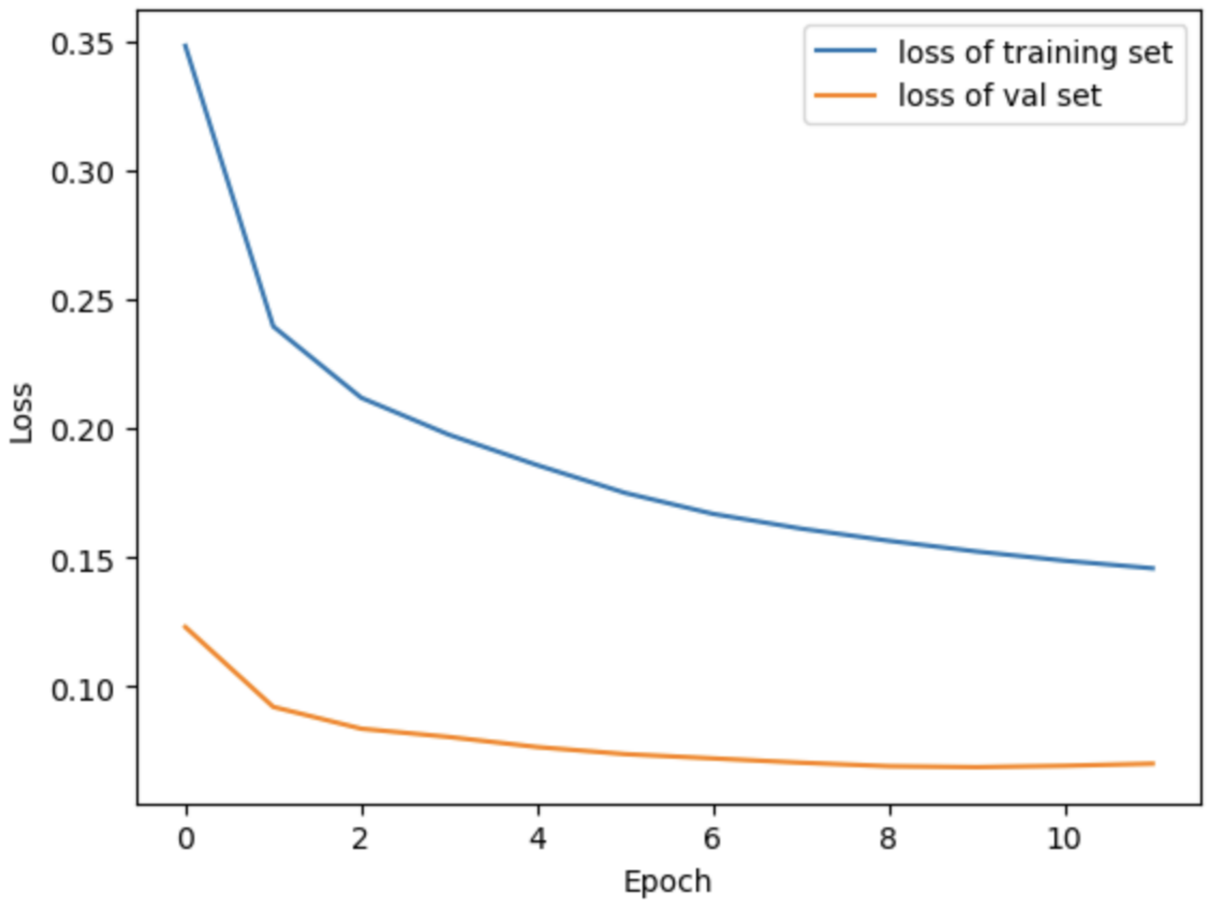
Then, I have divided my dataset into train_df[0:34575], val_df[34575:35575] and test_df[35575:]; and normalized the data using the formula given in the tutorial.

In the make_dataset, I changed shuffle to False.

Best model(Public score = 304.47705, Private score =): The best model scored by Kaggle for my model is 304.47705. To create this model, I have added bidirectional to all my LSTM layers. The LSTM layers has 128, 64 and 32 units and all of these LSTM has bidirectional added to them. Before implementing this model, I had expected this model to perform good as bidirectional processes in both directions, and it did worked well.

```
62/62 [==============================] - 0s 7ms/step - loss: 0.0699 - mean_absolute_error: 0.1845
```

The evaluation score on test set gave me a loss of 0.0699 and MAE of 0.1845.



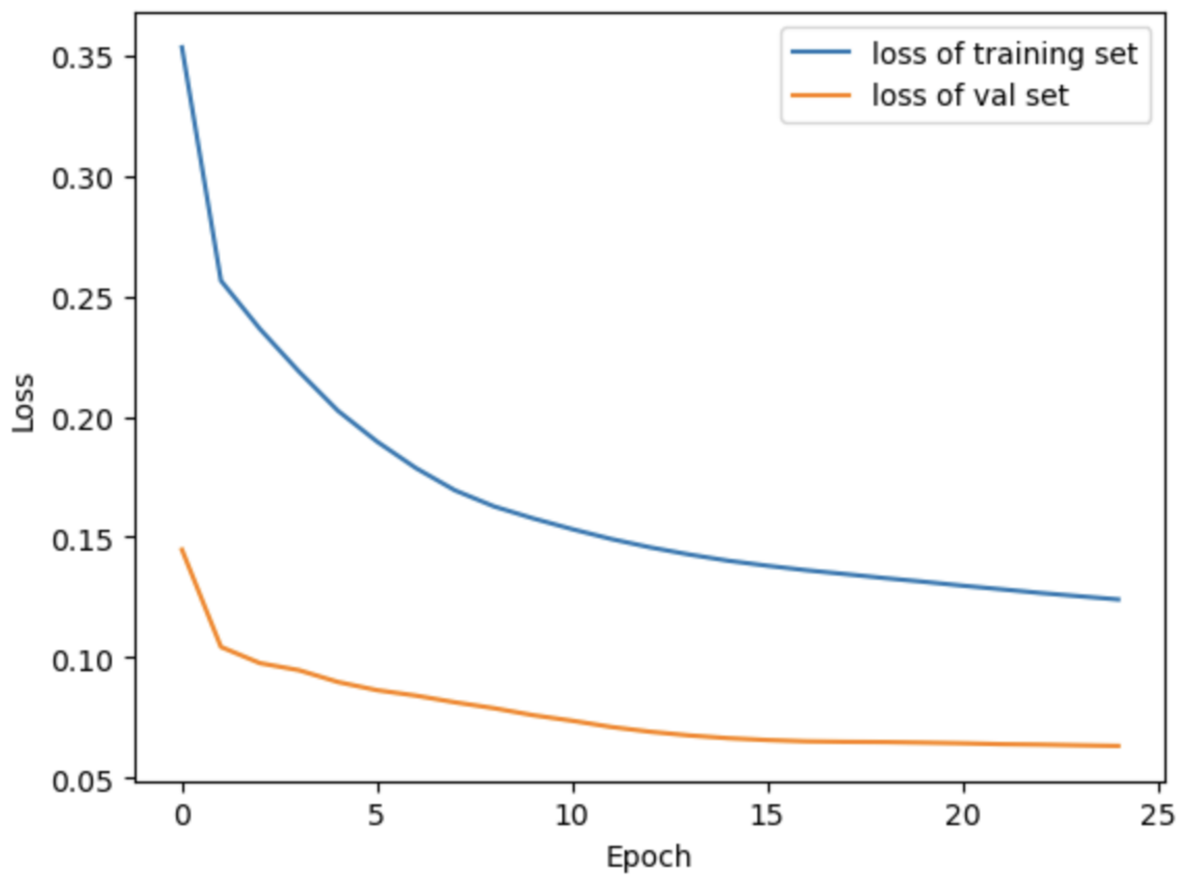
From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

Model 1: Vanilla LSTM Model

This model is same as in the TF tutorial with one LSTM layer having 32 recurrent units.

```
62/62 [==============================] - 0s 4ms/step - loss: 0.0632 - mean_absolute_error: 0.1724
```

The evaluation score on test set gave me a loss of 0.0632 and MAE of 0.1724.



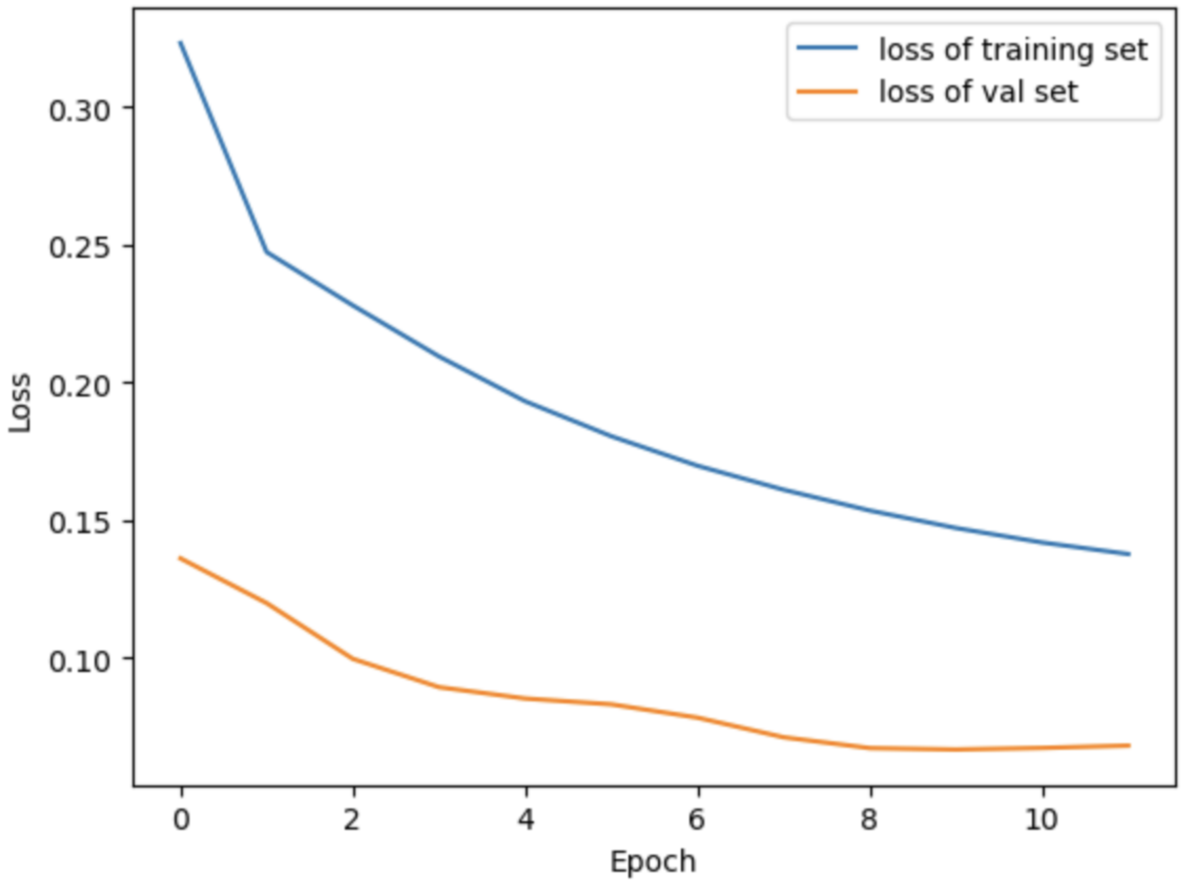
From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

Model 2:

Changed the number of recurrent units to the above model – 64.

```
62/62 [==============================] - 0s 4ms/step - loss: 0.0682 - mean_absolute_error: 0.1852
```

The evaluation score on test set gave me a loss of 0.0682 and MAE of 0.1852.

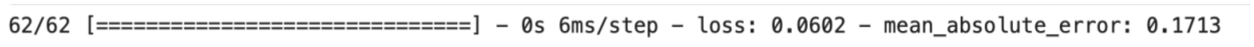


From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

Model 3:

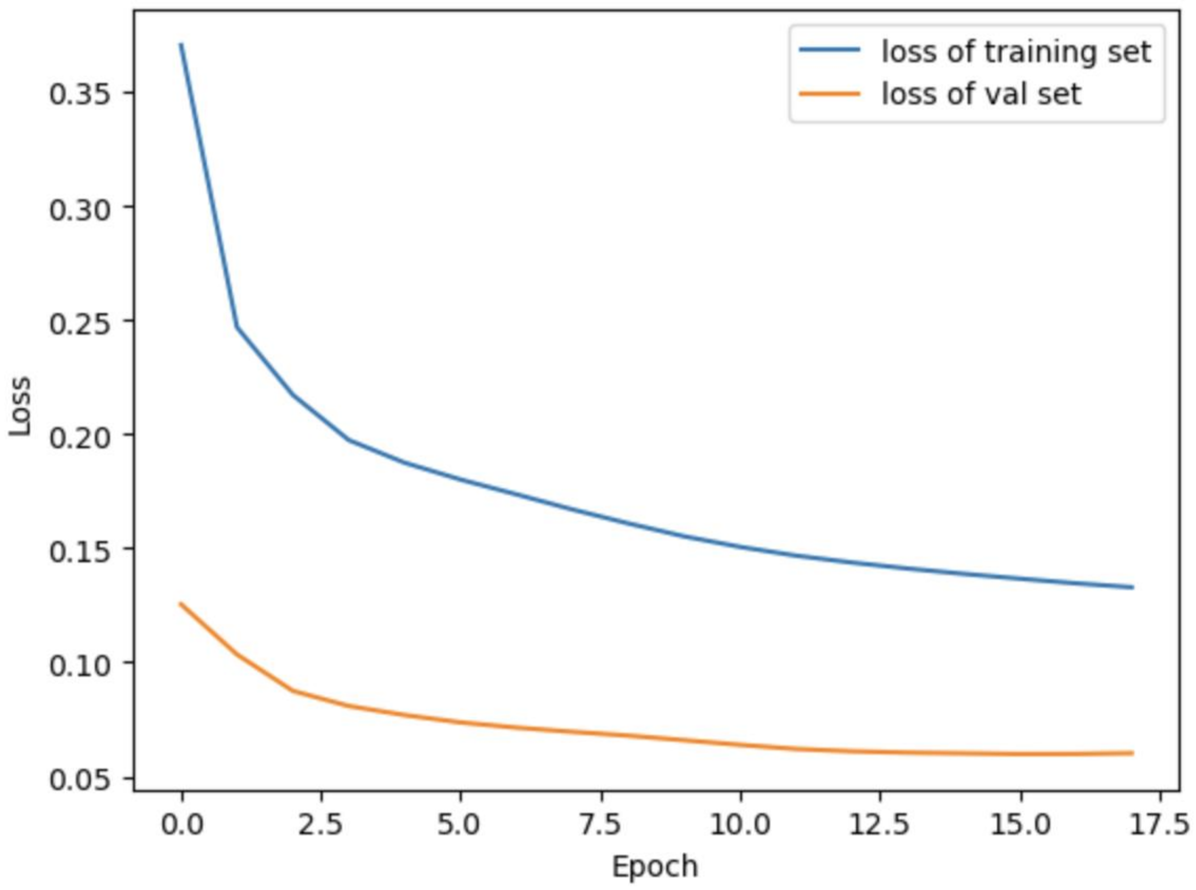
Adding the recurrent units and layers

Added 2 more layers with recurrent units of 128 and 64 respectively.



```
62/62 [==============================] - 0s 6ms/step - loss: 0.0602 - mean_absolute_error: 0.1713
```

The evaluation score on test set gave me a loss of 0.0602 and MAE of 0.1713.

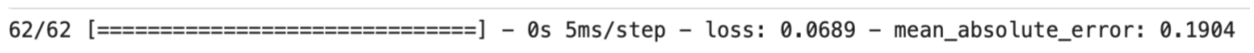


From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

<u>Model 4:</u>

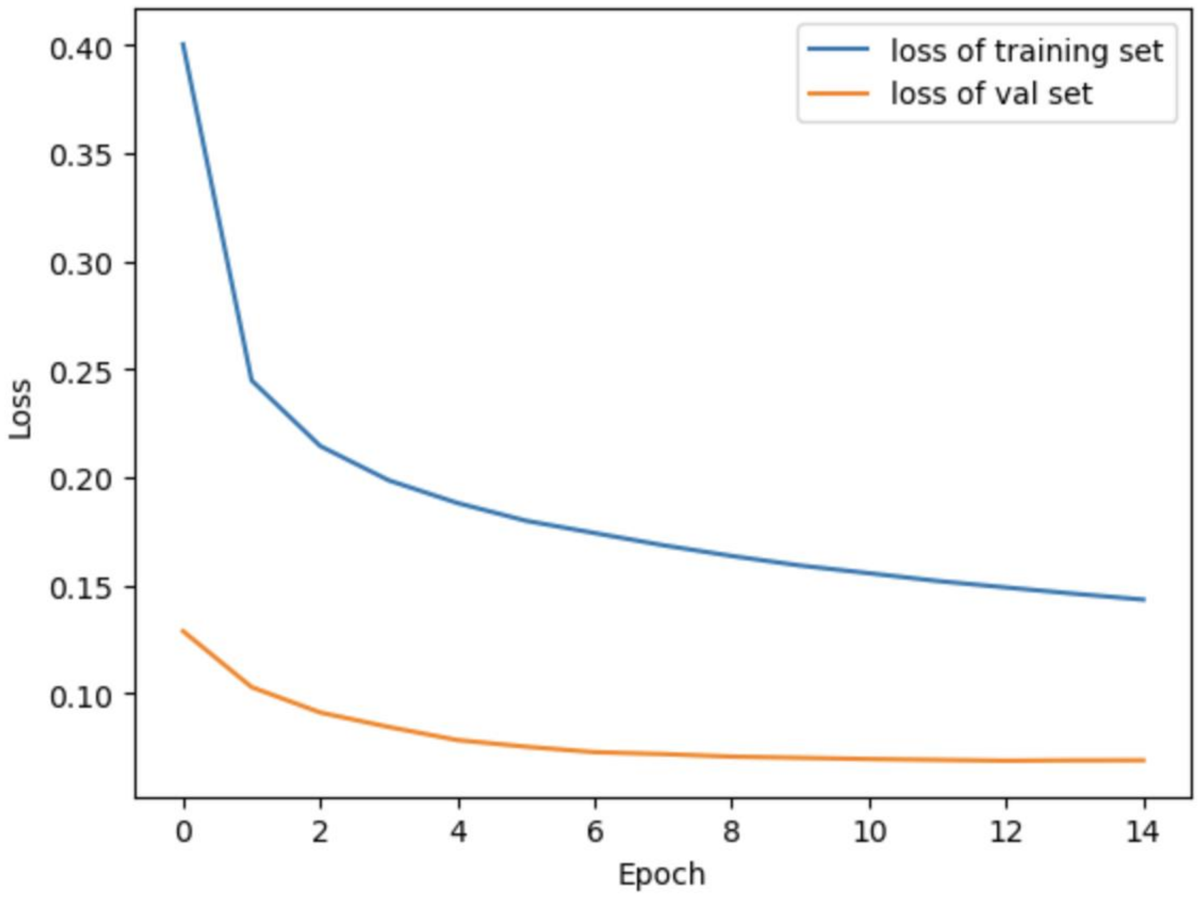
Adding stateful

Before adding stateful, I had expected the model to perform better, but it performed as same as the above models.



```
62/62 [==============================] - 0s 5ms/step - loss: 0.0689 - mean_absolute_error: 0.1904
```

The evaluation score on test set gave me a loss of 0.0689 and MAE of 0.1904.

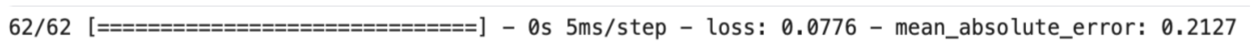


From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

Model 5:

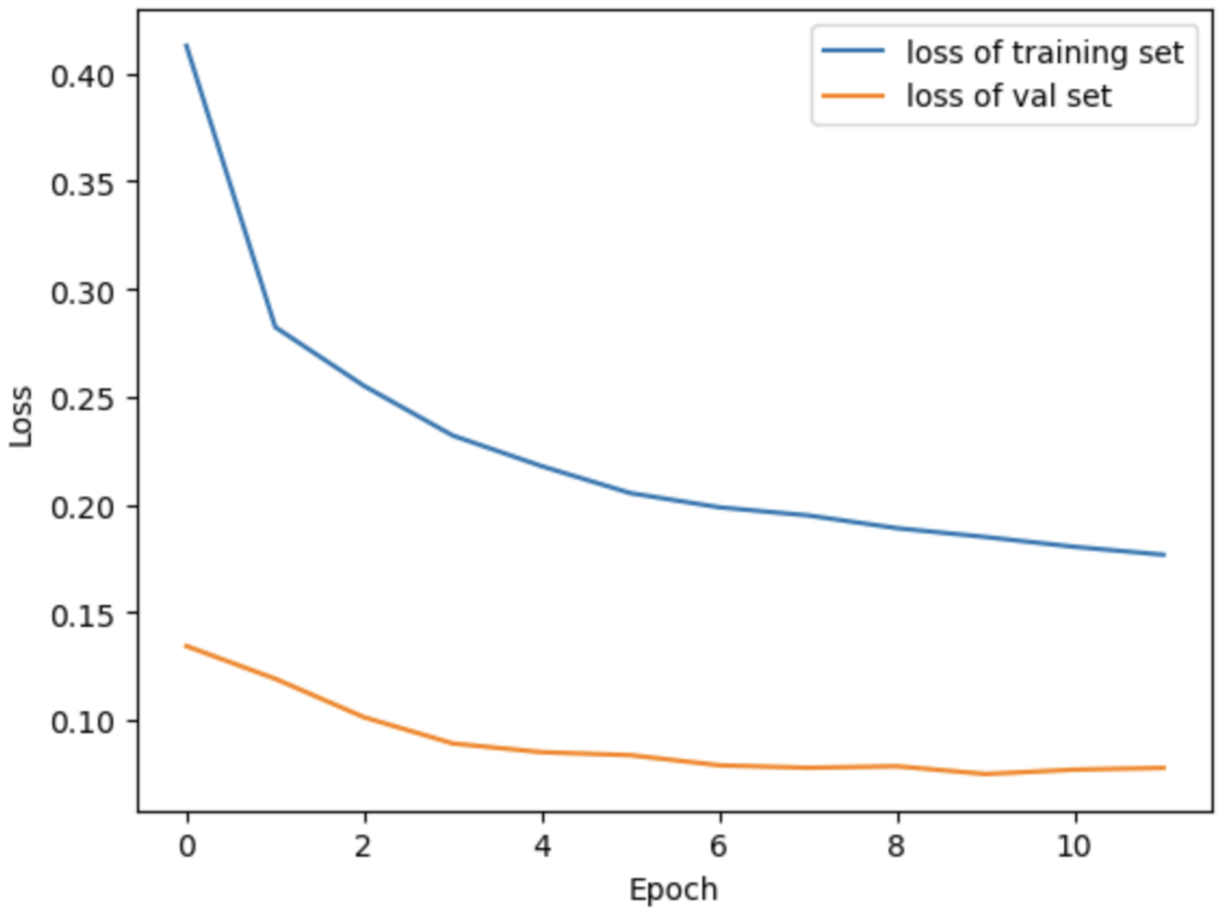
Adding dropout

As we know adding dropout helps reduce overfitting in the model, I had expected this to be my best model but Kaggle score for this model was in 500s, I suspect the reason to be that as the data is time series, implementing dropout on data did not help.



```
62/62 [==============================] - 0s 5ms/step - loss: 0.0776 - mean_absolute_error: 0.2127
```

The evaluation score on test set gave me a loss of 0.0776 and MAE of 0.2127.



From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

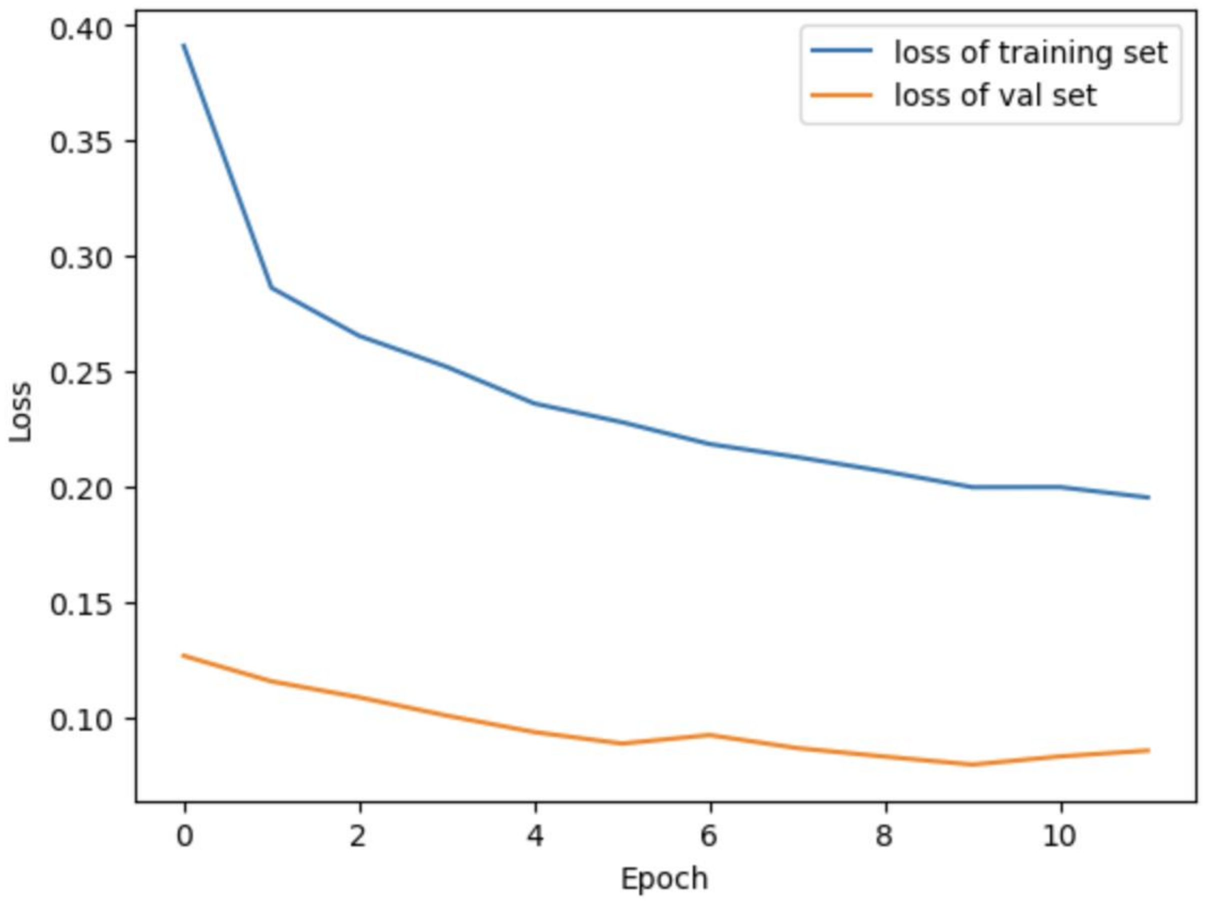
Model 6:

Adding dropout and recurrent dropout

Using both dropout and recurrent dropout together did not result in a better model. As my dropout model did not perform as good I expected the same performance and the results were quite similar to this model and with the dropout model.

```
62/62 [==============================] - 1s 9ms/step - loss: 0.0854 - mean_absolute_error: 0.2251
```

The evaluation score on test set gave me a loss of 0.0854 and MAE of 0.2251.



From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set, the loss of validation set kept decreasing and increasing over with the increase in the number of epochs.

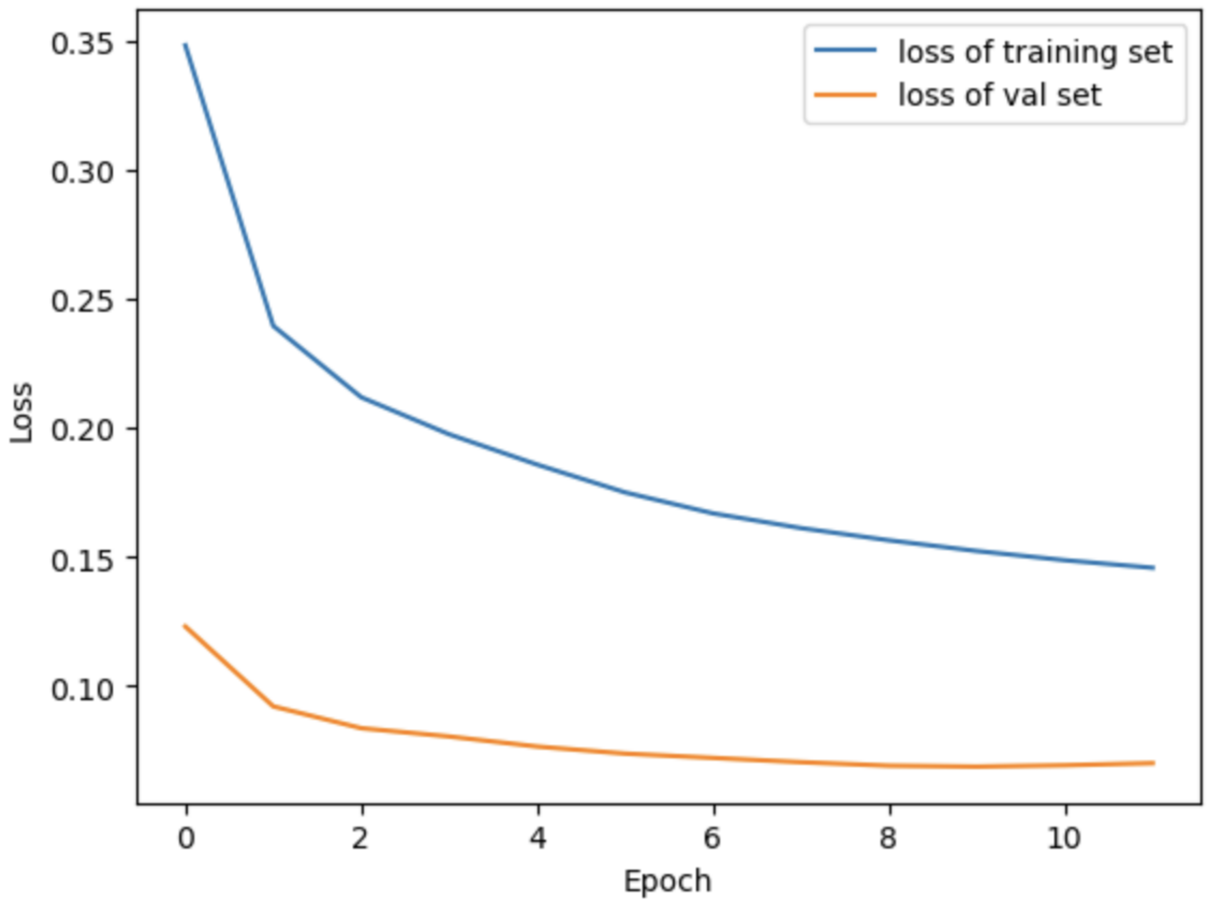
<u>Model 7</u>

Adding bidirectional RNN

The best model scored by Kaggle for my model is 304.47705. To create this model, I have added bidirectional to all my LSTM layers. The LSTM layers has 128, 64 and 32 units and all of these LSTM has bidirectional added to them. Before implementing this model, I had expected this model to perform good as bidirectional processes in both directions, and it did worked well.

```
62/62 [==============================] - 0s 7ms/step - loss: 0.0699 - mean_absolute_error: 0.1845
```

The evaluation score on test set gave me a loss of 0.0699 and MAE of 0.1845.



From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set, but also we need to consider the fact that the loss of validation set started from lesser value.

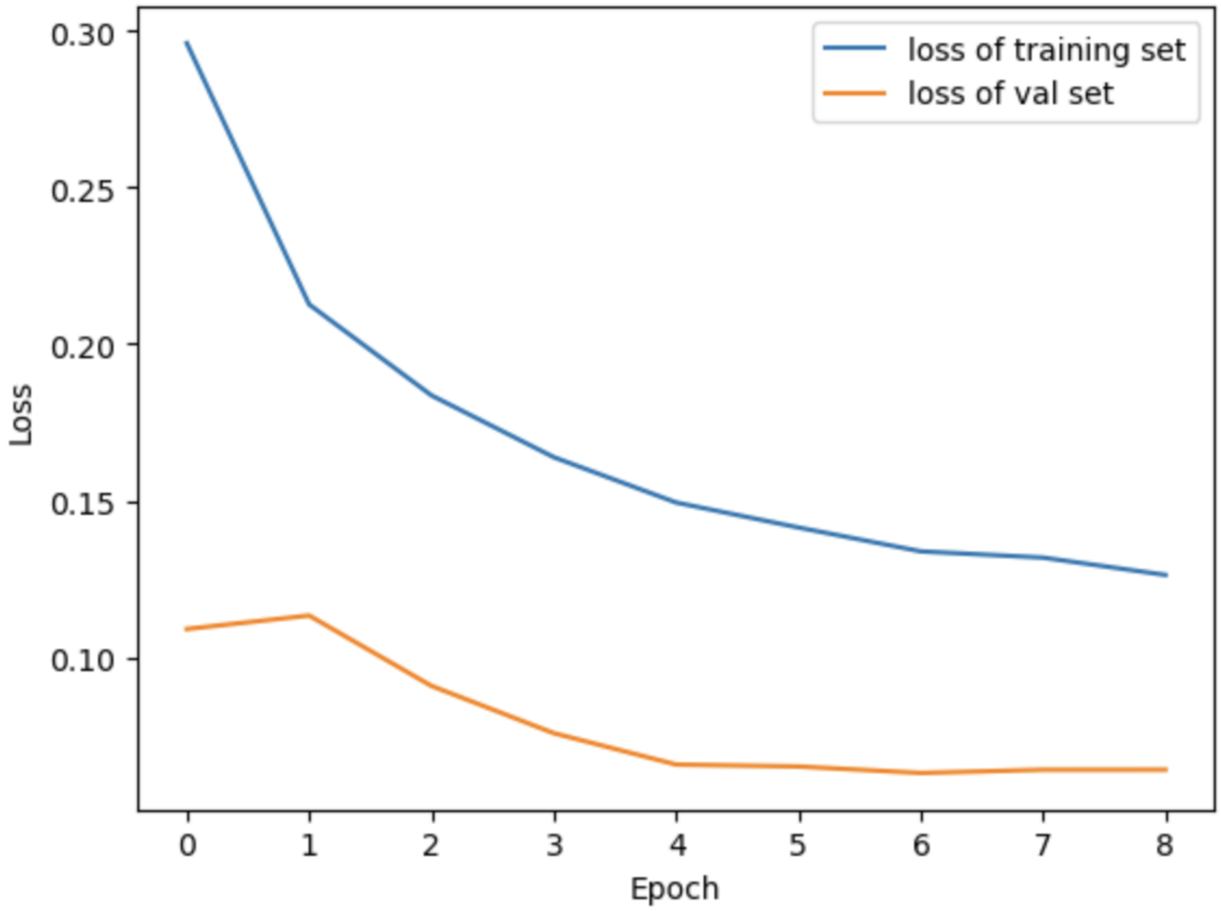
<u>Model 8</u>

Adding Conv1D with RNN

Implementing Conv1D with my RNN model sure did perform good, but did not perform good with kaggle's metrics calculation.

```
62/62 [==============================] - 1s 7ms/step - loss: 0.0641 - mean_absolute_error: 0.1929
```

The evaluation score on test set gave me a loss of 0.0641 and MAE of 0.1929.



From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set.

<u>Model 9/ Runner-up model</u>

Adding GRU with Conv1D

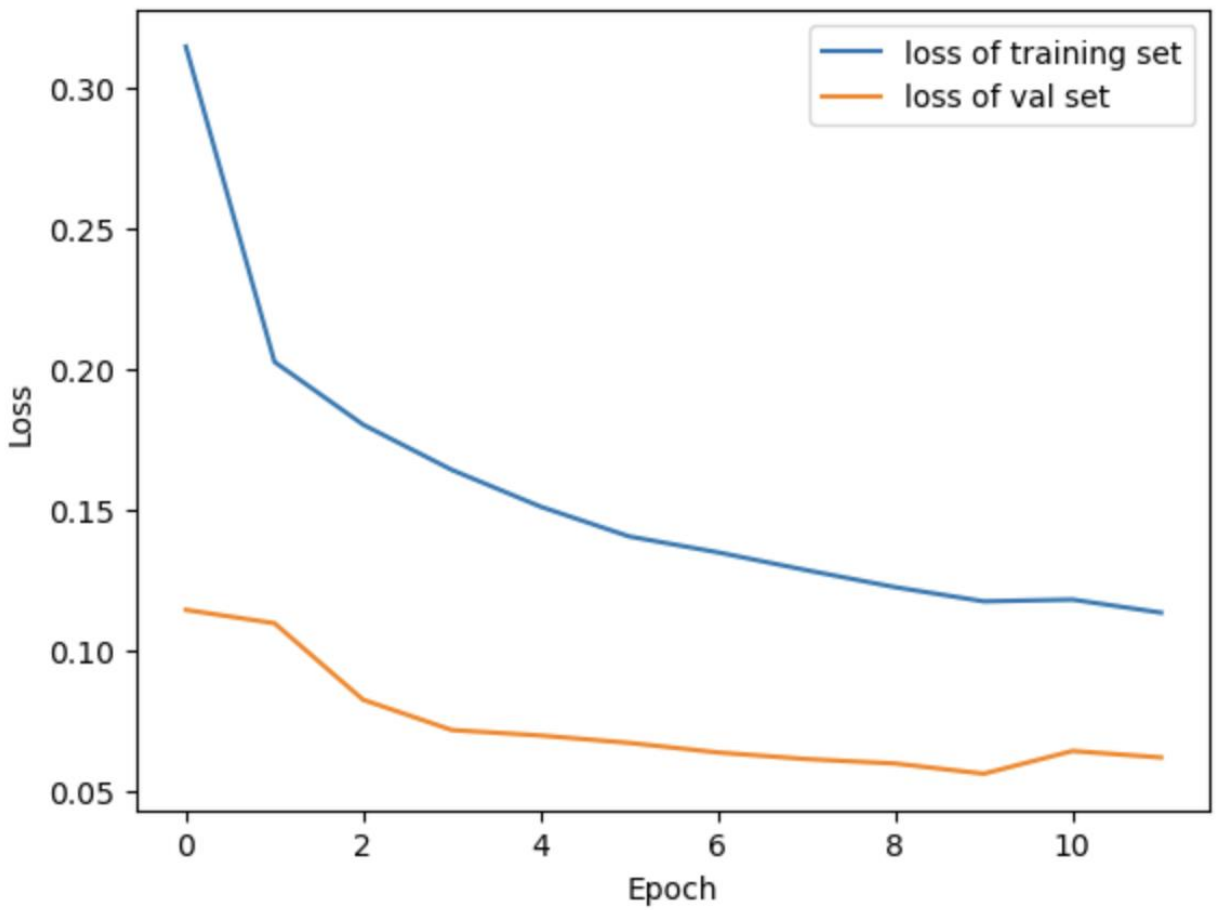
As conv1D is usually applied to sequential data and GRU for time series analysis, implementing a model together will result in better capturing of local patterns or features before GRU comes in to play.

Together they are used to extract features and capturing patterns.

Before implementing this model I had expected it to be my best model, but it did not perform that well.

```
62/62 [==============================] - 0s 6ms/step - loss: 0.0618 - mean_absolute_error: 0.1699
```

The evaluation score on test set gave me a loss of 0.0618 and MAE of 0.1699.



From the graph we can see that as the number of epochs increased, the loss of training set decreased more that compared to the validation set, as loss of validation set remained quite same for 2 epochs and decreased and then increased again after epoch 9.

According to me, I will declare this model as my runner up model. As with 2 types of RNN architectures being implemented together, the loss and MAE managed to be less, similar to my best model. Also, as mentioned above this model is used to extract features and capture trends, it will be really helpful to identify trends.

Conclusion on best model:

With only three layers of LSTM with bidirectional applied the model gave me the best results overall. The loss was 0.069 and MAE was 0.1845, and also less layers means less time-consuming.

Reaction and reflection:

On one hand implementing different models, adding and removing parameters, layers, recurrent units, dropout, recurrent dropout was really fun; but waiting for those models to run and get the results and make submissions to Kaggle required some real patience and was time consuming. Overall, I enjoyed working on this project.