ASSIGNMENT 3

WEATHER TIME SERIES FORECASTING

Bhumika Shah

811286091

Introduction

Let's break down the key differences between time series data and other types of datasets, using a temperature forecasting exercise as an example. RNNs have been particularly suited for such tasks because they can capture the sequential nature of such data better than any convolutional or densely connected network. In our experiments, we use 50% of the data for training and 25% each for validation and testing. Since we are predicting temperatures of the future based on past data, it is important to use newer data for validation and testing compared to the training data. The problem we are trying to solve is: Can we predict the temperature for a day using the hourly temperature data from the past five days? In preparing this data, vectorization wasn't needed because it was already in numerical form.

We tested 14 different models in order to find the best model for time series data. Our simple model baseline gave us a Mean Absolute Error of 2.62, using very common-sense methods. The Simple Neural Network Model used a dense layer; it has an MAE of 2.65 since it lost valuable temporal information by flattening the data. A Convolutional model also did not yield much success since the nature of the sequence could not be extracted this way.

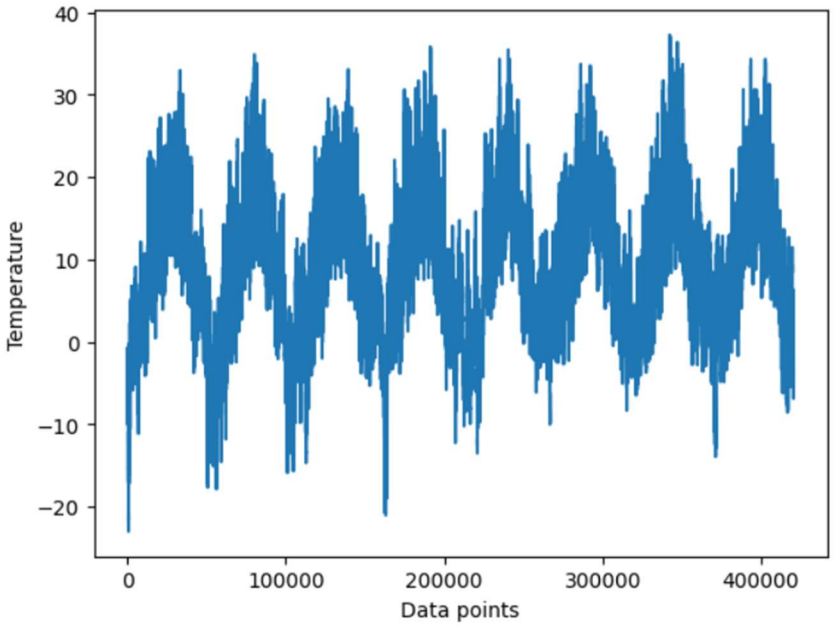
These results steered us towards RNNs, more fitted for the time series, since they are based on information from previous steps to predict. It is this "memory" in RNNs that allows them to learn patterns across time. However, basic RNNs often suffer from the problem of a "vanishing gradient", which makes it hard for them to learn patterns over long horizons in deep networks. For this problem, we implemented more complicated variants of RNN, such as the Gated Recurrent Unit and Long Short-Term Memory networks available in Keras.

Among all models, the GRU model performed the best since it is computationally more efficient compared with LSTM but also captured the long-range patterns of the data. When testing various models on different LSTM models with different units, such as 8, 16, and 32 units, the best-performing model used 8 units. LSTM is one of the most popular models when it comes to time series data; thus, we improved its performance by incorporating some techniques, such as bidirectional input and recurrent dropout to prevent overfitting. These LSTM models consistently reported a lower MAE than the baseline.

Finally, we tried the RNN and 1D convolution hybrid model, which increased the MAE to 3.95, probably because convolution disrupts the sequential order of data.

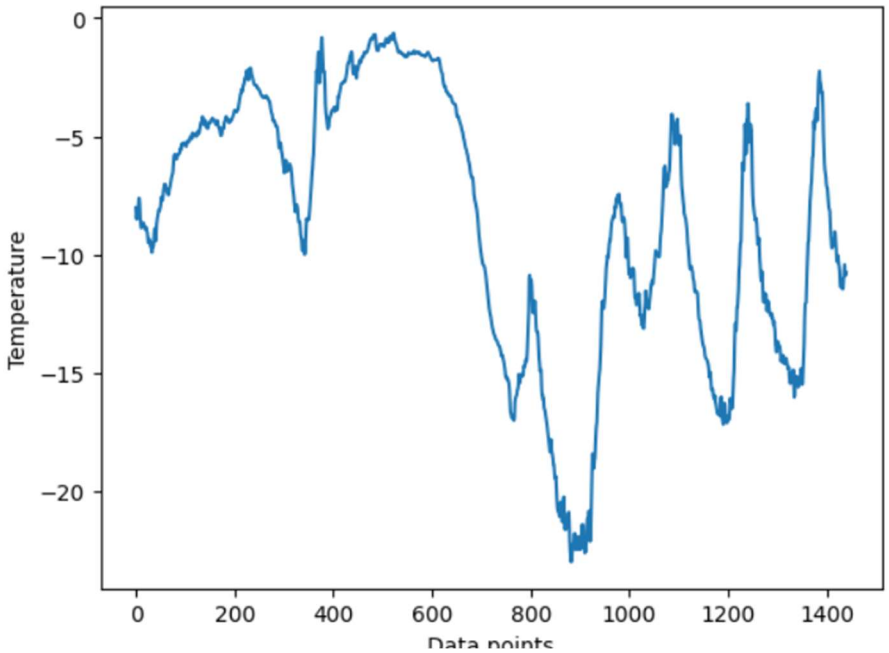
In short, simple RNNs are not good for time series analysis due to vanishing gradients and an inability to keep track of long-term memory. The LSTMs and GRUs would be perfect for the task, although GRUs alone are way more efficient computationally in most scenarios. The model can further be improved by tuning hyperparameters regarding the number of units, dropout rates, and whether to configure it as bidirectional. In the case of time series data, it is preferable to use RNN architectures instead of convolutional methods, which generally lose information about the sequence, sometimes necessary for an accurate prediction.

Seasonal Temperature Variations Over Time:



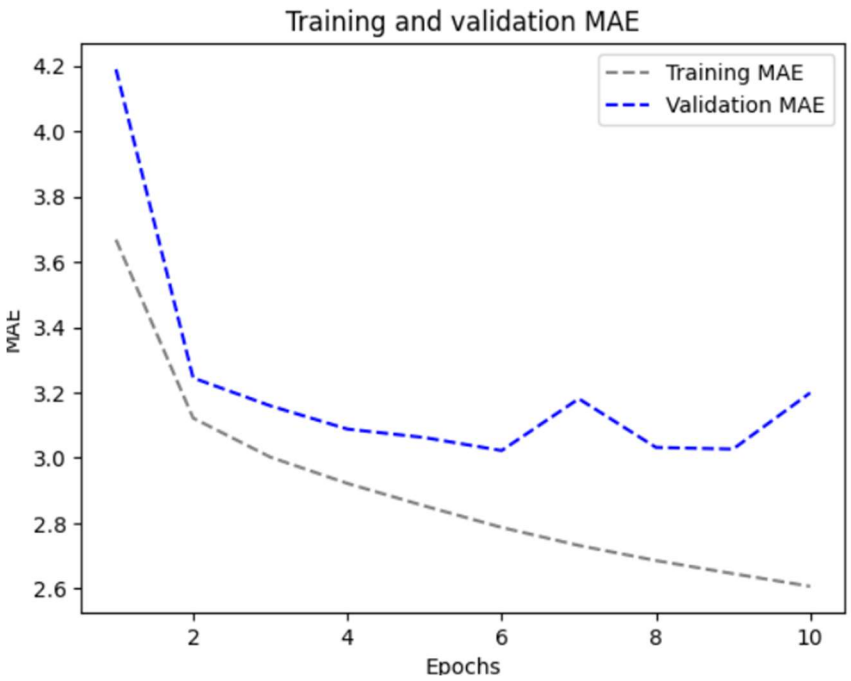
There is a well-marked seasonal pattern with high temperatures in the middle of the year and low temperatures at the start and end of the year. The x-axis data points indicate an extended observation period.

Time Series of Temperature:



Dense layer in Machine learning Model:

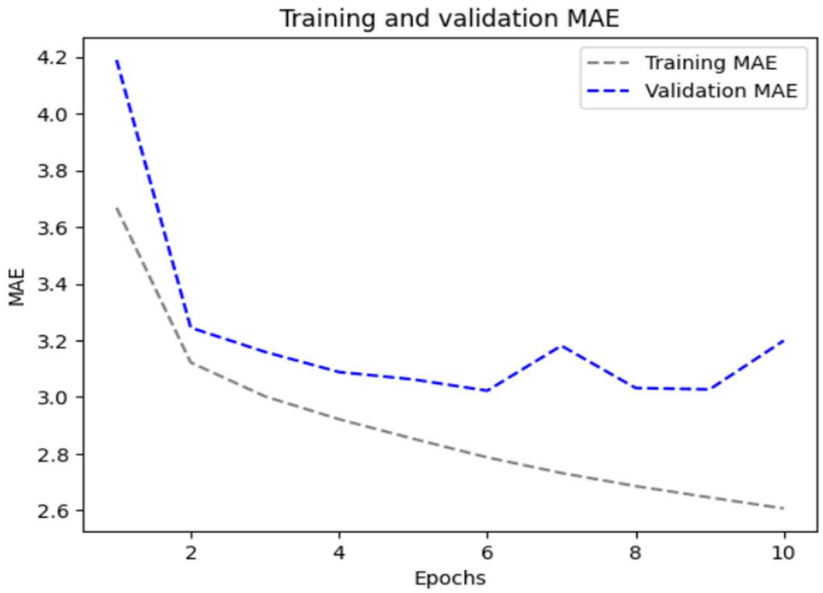
```
405/405 [==============================] - 3s 7ms/step - loss: 11.3070 - mae: 2.6472
Test MAE: 2.65
```



1D convolutional model:

This may be appropriate for daily cycles of temperature since the convolutional model can make good use of the fact that similar patterns occur over different days in much the same way that a spatial convolutional network makes good use of patterns occurring in different parts of an image. However, this kind of temporal convolutional network doesn't perform as well for meteorological data as a densely connected model. That is because most weather data don't meet the condition of translation invariance. Consequently, this yielded an MAE in validation of about 3.26 degrees, way higher than some reasonable baseline we were looking for.

```
405/405 [==============================] - 3s 7ms/step - loss: 17.0799 - mae: 3.2569
Test MAE: 3.26
```



A Simple RNN:

RNNs are particularly good at finding the intricate patterns of sequences because they take into consideration previous time steps to inform their decisions in the present time. This means that an RNN inherently has "memory" and can process sequences of variable lengths. In many ways, this is a theoretical expectation of any RNN model, which retains all previous inputs. However, in practice, there are many obstacles with basic RNNs, especially when training deep ones. Our results showed very well that the simplest RNN model turned out to be the worst performer among all the models tried. Advanced types of RNNs, including LSTM and GRU implemented in Keras, are explicitly designed to cope with all these problems.

```
405/405 [==============================] - 4s 9ms/step - loss: 151.2857 - mae: 9.9179
Test MAE: 9.92
```

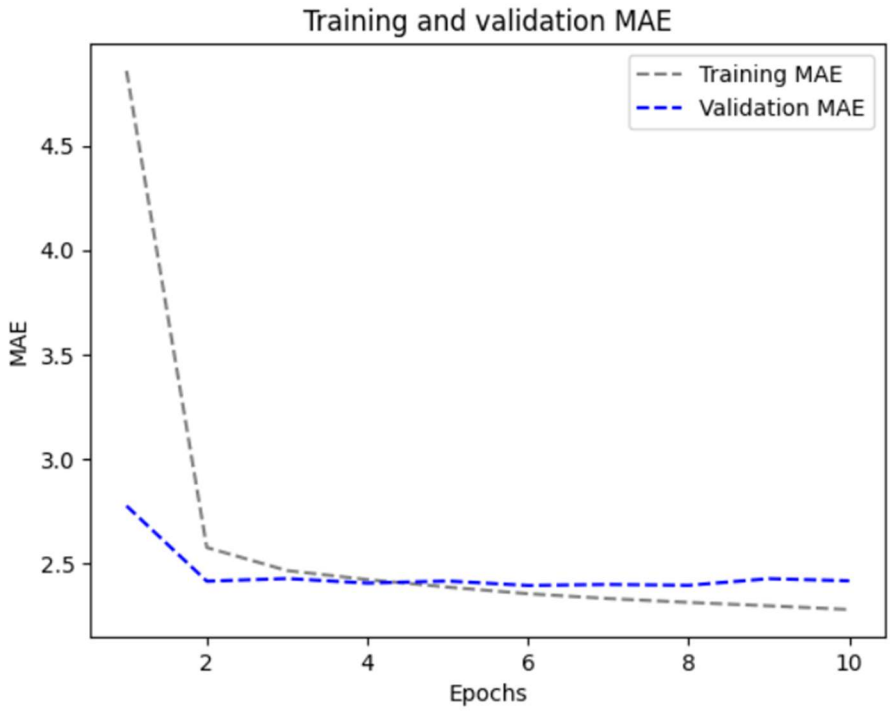
Staking RNN layers

```
405/405 [==============================] - 8s 20ms/step - loss: 151.1067 - mae: 9.900
3
Test MAE: 9.90
```

A Simple Gated Recurrent Unit (GRU):

We will use GRU layers instead of LSTM layers because GRU is a lighter version of the LSTM. In our tests, the GRU model was the most efficient one in giving the lowest Test MAE of 2.57. Besides, it consumes less computational power than LSTM while still being capable of capturing long-run patterns of sequential data.

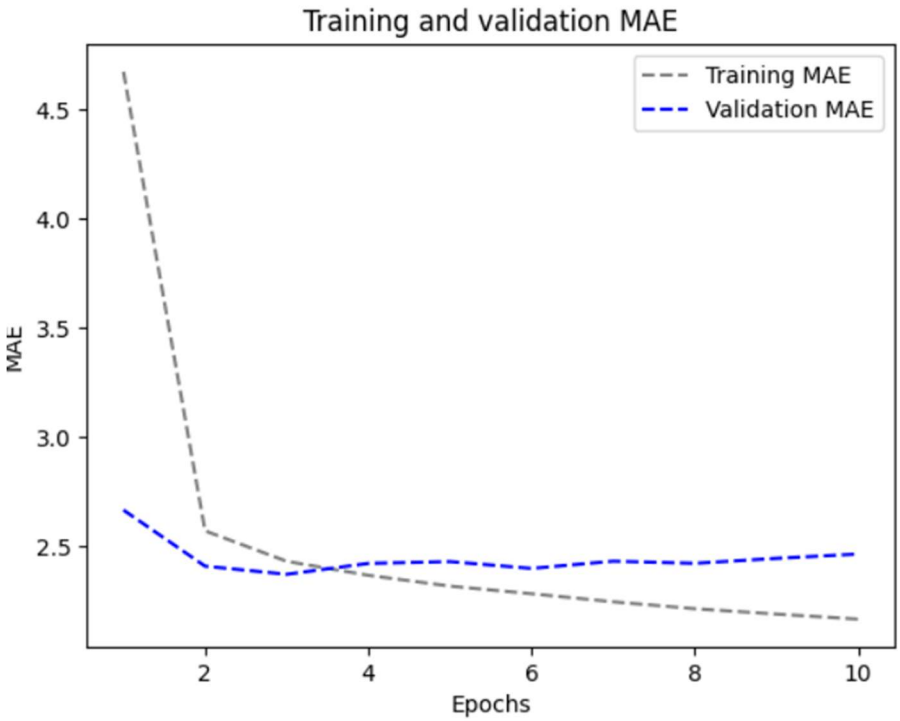
```
405/405 [==============================] - 7s 17ms/step - loss: 10.8892 - mae: 2.5670
Test MAE: 2.57
```



LSTM Simple:

The LSTM-based model achieved a validation MAE of 2.46 degrees and a test MAE of 2.53 degrees. This shows the effectiveness of machine learning in this task, with the model slightly outperforming the common-sense baseline, at least for now.

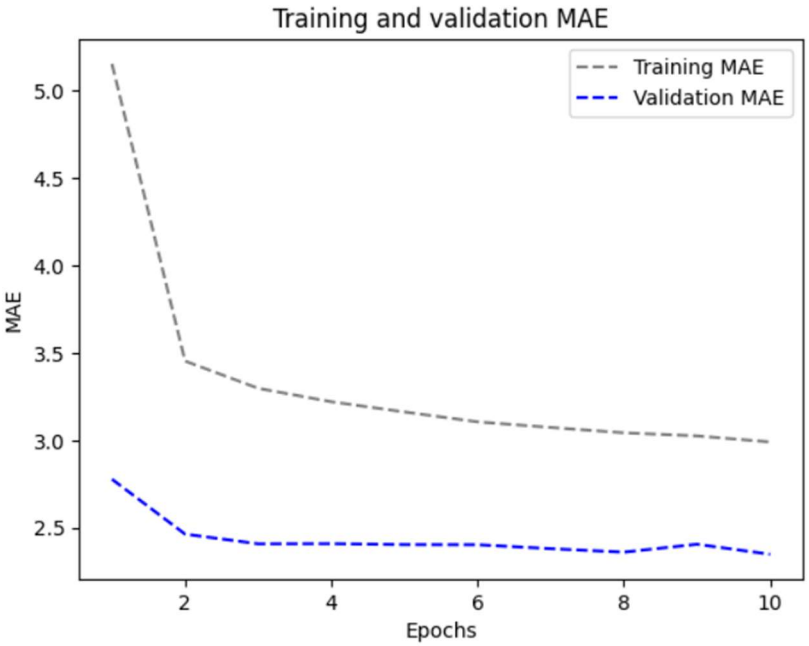
```
405/405 [==============================] - 9s 21ms/step - loss: 10.4407 - mae: 2.5350
Test MAE: 2.53
```



Dropout Regularization:

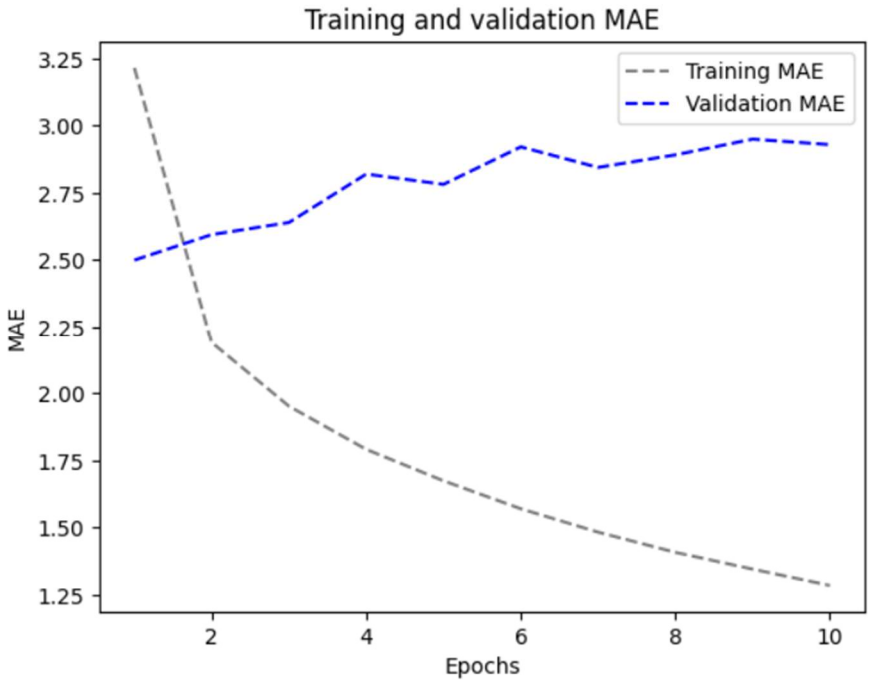
We have overcome the overfitting that was occurring within the initial five epochs. Our validation MAE is at 2.34 degrees and the test MAE is at 2.61 degrees, which is pretty awesome. I was able to build six different LSTM models by using 8, 16, and 32 units in the stacked recurrent layers.

```
405/405 [==============================] - 10s 23ms/step - loss: 11.4842 - mae: 2.612
6
Test MAE: 2.61
```



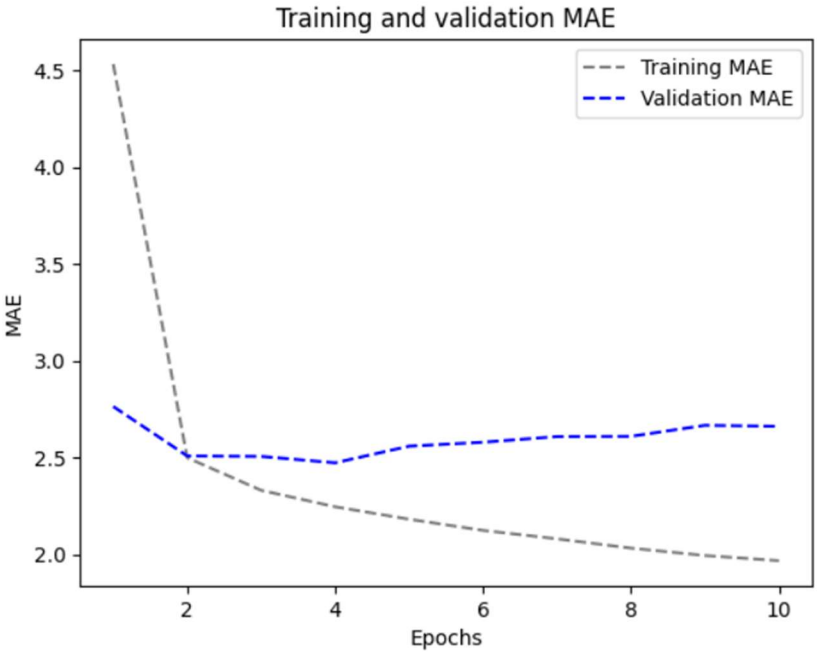
Stacked Setup with 32 units:

```
405/405 [==============================] - 26s 62ms/step - loss: 12.3380 - mae: 2.76
0
Test MAE: 2.76
```



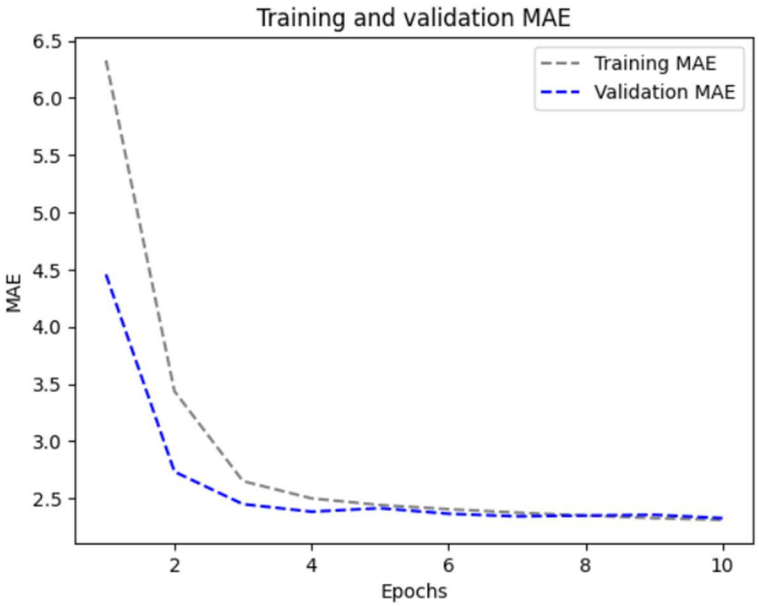
Stacked setup with 16 units:

```
405/405 [==============================] - 18s 41ms/step - loss: 11.2779 - mae: 2.657
5
Test MAE: 2.66
```



Stacked setup with 8 units:

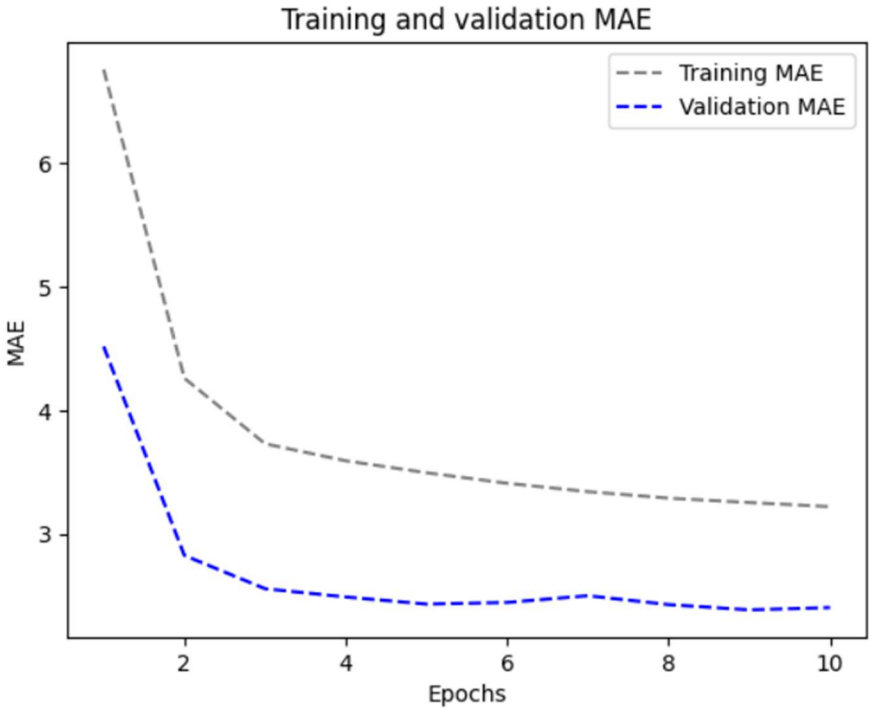
```
405/405 [==============================] - 12s 28ms/step - loss: 10.1453 - mae: 2.485
9
Test MAE: 2.49
```



The best model had 8 units and gave an MAE score of 2.49. Other fine-tuning included the use of techniques like bidirectional data, whereby information fed into a recurrent network is in multiple formats to lower MAE, or adding recurrent dropout to help with overfitting. These changes resulted in similar MAE values for the models, and strikingly, all consistently below that from the common-sense model, as shown in the graph of the MAE assessment.

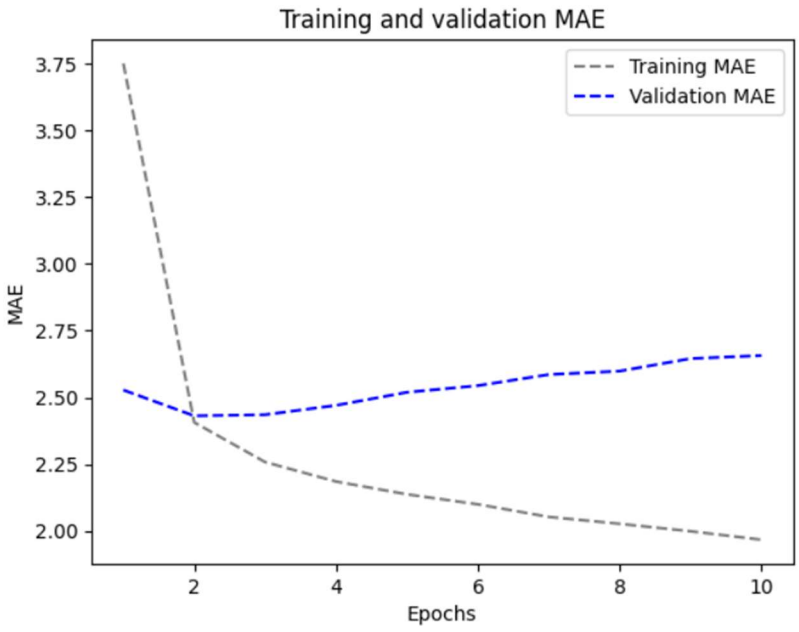
Dropout-regularized, stacked model (LSTM)

```
405/405 [==============================] - 14s 33ms/step - loss: 10.7908 - mae: 2.548
1
Test MAE: 2.55
```



Bidirectional Long Short-Term Memory

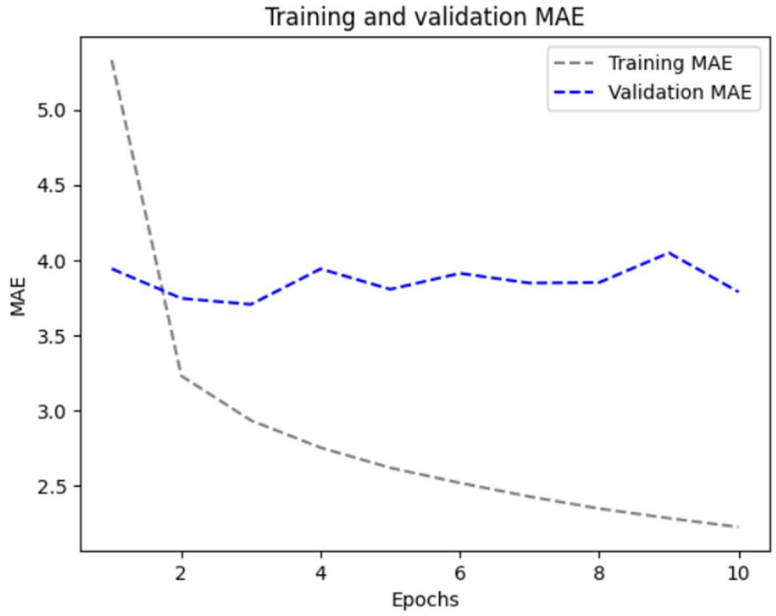
```
405/405 [==============================] - 11s 24ms/step - loss: 10.2981 - mae: 2.528
3
Test MAE: 2.53
```



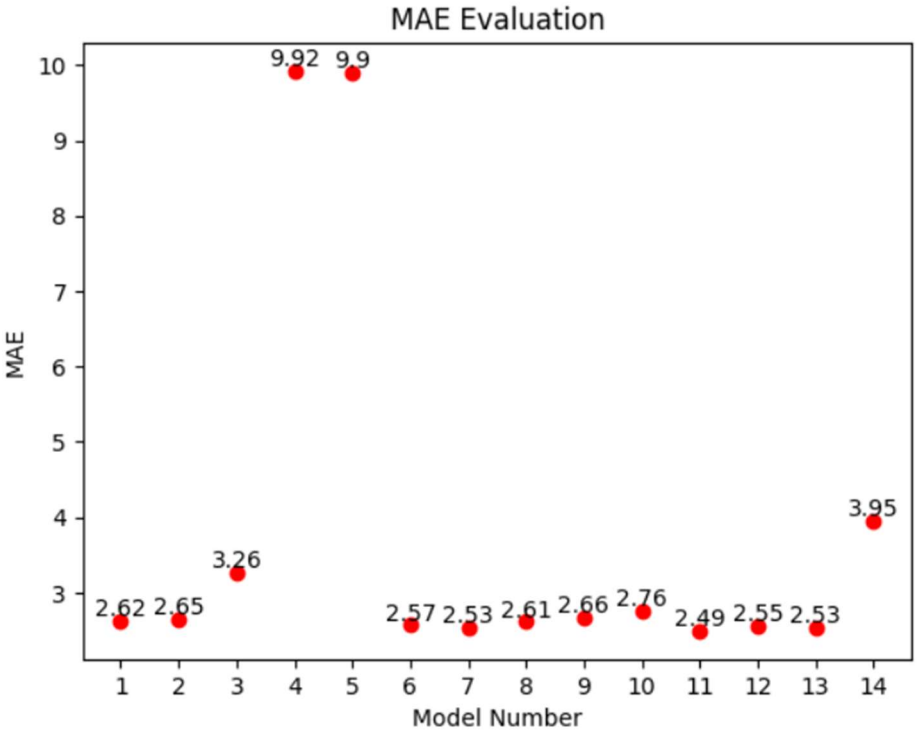
Combining 1D Convolutional Networks and LSTM

```
405/405 [==============================] - 4s 7ms/step - loss: 24.4383 - mae: 3.9546
Test MAE: 3.95
```

The RNN and 1D convolution model I made did not do a very good job, reaching only an MAE of 3.95. Perhaps this might be due to the fact that the convolution itself disrupts the order of the information.



MAE Evaluation



Model Number	MAE
1	2.62
2	2.65
3	3.26
4	9.92
5	9.9
6	2.57
7	2.53
8	2.61
9	2.66
10	2.76
11	2.49
12	2.55
13	2.53
14	3.95

In short, my best models are the advanced RNNs including LSTM and GRU; the model combining RNN with 1D convolution worked worse. Although the Bidirectional LSTM is still widely used, my test showed that GRU was more efficient in time series data. More modifications such as the number of units in stacked layers, dropout rates, and using bidirectional data can be performed to further optimize GRU.