**SUMMARY**

We created a total of 14 models for time series data analysis. As a baseline, the initial model produced a Mean Absolute Error (MAE) of 2.62 and depended on common sense techniques. The MAE of 2.70 was marginally higher after we developed a simple machine learning model with a thick layer. The time series data was flattened, eliminating the temporal context, which led to poor performance of the dense layer model. Convolutional models were also tried, but the results were not good since the model handled all the data segments equally—even after pooling—disturbing the sequential order of the data.

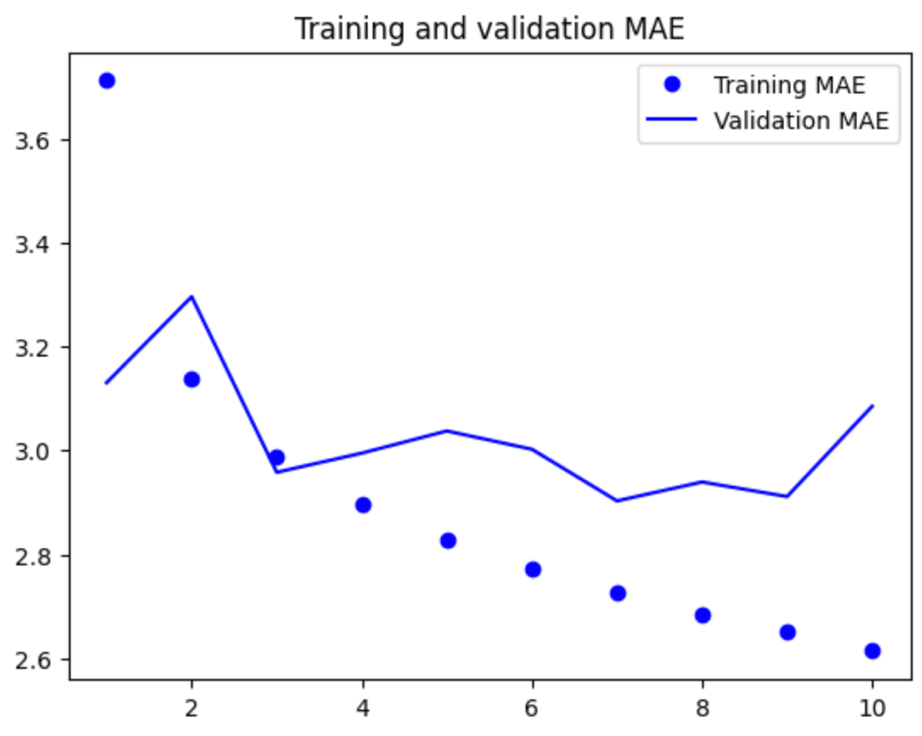
We concluded as a result that Recurrent Neural Networks (RNNs) perform better with time series data. Recurrent neural networks' (RNNs') ability to integrate knowledge from previous stages into current decision-making is a crucial component. The network may then use this to find patterns and dependencies in the sequential data. The RNN can represent sequences of different lengths because of its internal state, which functions as a kind of memory and retains data from previous inputs. But frequently, the fundamental Simple RNN is too straightforward to be very useful.

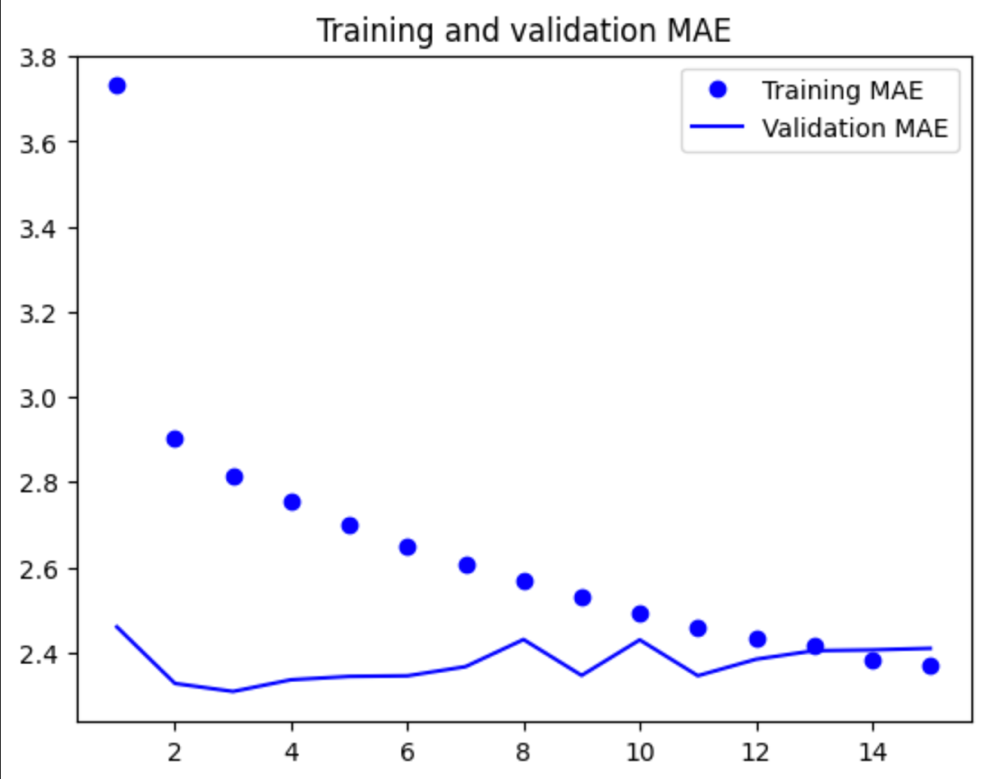
The graphical depiction indicates that Simple RNN consistently performs the worst out of all the models, which is a notable downside. Although the renowned "vanishing gradient problem" causes Simple RNN to struggle operationally, particularly in deep networks, it should be able to retain knowledge from all prior time steps theoretically. The network is essentially untrainable because to this issue. More sophisticated RNN variations were created in response to this difficulty and are incorporated into Keras as the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). The simplest GRU model produced the greatest results from our experiments out of all the models, mostly because it is more computationally efficient than LSTMs and can capture long-range relationships in sequential data.

LSTMs are a well-known architecture for processing time series data efficiently. We tested six different LSTM models with varied numbers of units in stacked recurrent layers (8, 16, and 32), and the model with eight units performed the best. Recurrent dropout was also used to avoid overfitting, and bidirectional data presentation was tested to improve accuracy and solve the forgetting issue. The MAE values of these LSTM models were all comparable and consistently less than those of the common sense model.

Finally, we tried to integrate an RNN with a 1D convolution model. The hybrid model produced a greater mean absolute error (MAE) of 3.79, which can be attributed to the limits of the convolution in preserving the information order. My findings suggest that basic RNNs should be avoided for time series analysis since they have trouble with the vanishing gradient issue and are unable to accurately capture long-term relationships. Instead, take into account more sophisticated RNN designs that are intended to get around these obstacles, including LSTM and GRU. Although GRU may provide more effective outcomes than LSTM, our trials indicate that LSTM is a popular option for processing time series data.

**FIGURES:**

****

****

Hyperparameters like the number of units in stacked recurrent layers, recurrent dropout rates, and the usage of bidirectional data presentation may all be tuned to improve GRU models. Additionally, since the combination of RNN with 1D convolution did not provide the best results, it is advised to concentrate on RNN designs designed for sequential data. Convolutional methods are less appropriate for time series data processing since they frequently cause information to be out of order.