ASSIGNMENT-3

WEATHER TIME SERIES FORCASTING

SUMMARY

Using this temperature-forecasting exercise, we will demonstrate the primary distinctions between time series data and the various dataset types we have previously worked with. We will also demonstrate RNNs, a new kind of neural network. Convolutional and heavily connected networks cannot manage this type of dataset, yet machine learning techniques are completely adept at handling it. 50% of the data will be used for training, 25% for validation, and the remaining 25% for testing in all of our investigations. Since our goal is to forecast the future based on the past rather than the reverse, validation and test data used with time series data must be more recent than the training data. The validation/test divides ought to reflect this. The problem will be stated exactly as follows: Can information collected hourly for the preceding five days be used to determine the temperature on a particular day?

Let's start by preparing the data so that a neural network may be trained to use it.
It's easier said than done—vectorization is not required because the data is numerical in the first place.We developed 14 distinct models to analyze time series data. The first model generated a baseline and a Mean Absolute Error (MAE) of 2.62 using commonsense methods. Following that, we created a straightforward machine learning model with a dense layer, which resulted in a somewhat higher MAE of 2.65.The dense layer model performed poorly as a result of the time series data being flattened, which removed the temporal context. A convolutional model was also employed, however it yielded poor results since it disturbed the data's sequential order by treating each data segment identically, even after pooling.

As a result, we came to the conclusion that time series data is better suited for Recurrent Neural Networks (RNNs). One essential feature of recurrent neural networks (RNNs) is their capacity to incorporate information from earlier stages into the decision-making process at hand. This can then be used by the network to identify dependencies and patterns in the sequential data. Because of its internal state, which acts as a sort of memory and stores information from prior inputs, the RNN may represent sequences of varying lengths.

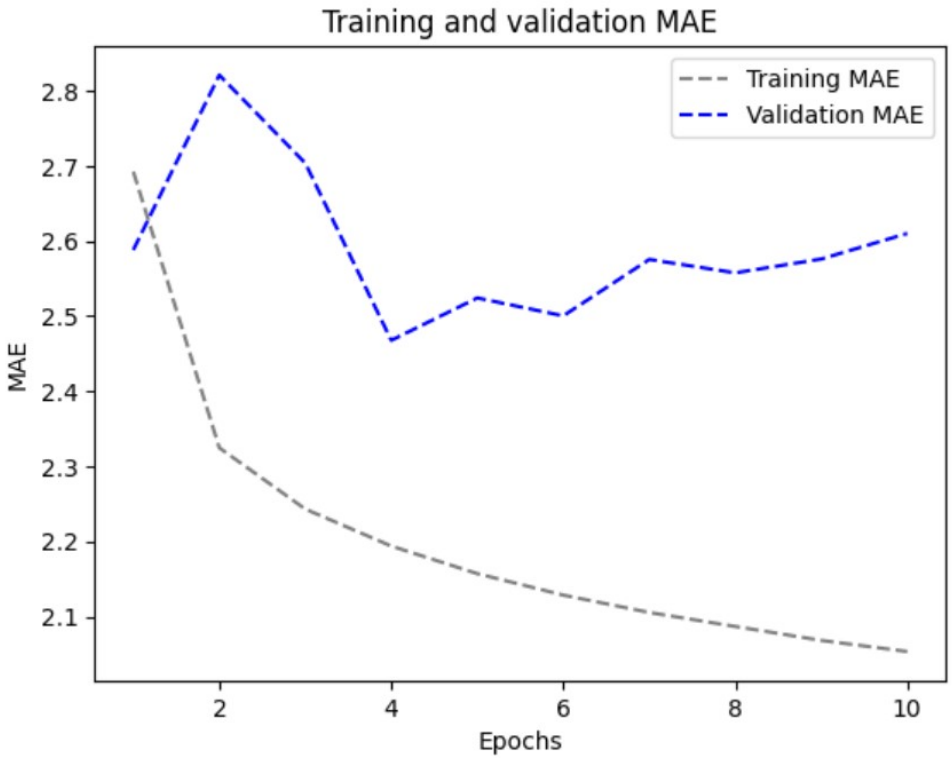
However, the basic Simple RNN is frequently too simple to be of any service. One significant drawback of Simple RNN is that, according to the graphical representation, it consistently performs the worst of all the models. While the well-known "vanishing gradient problem" makes Simple RNN difficult to employ in practice, especially in deep networks, it should theoretically be able to remember information from every previous time step. Due to this problem, the network is practically untrainable. In response to this challenge, more complex RNN variants were developed and integrated into Keras as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Out of all the models, the most straight forward GRU model yielded the best results from our tests, mostly because to its higher computational efficient than LSTMs, the most straightforward GRU model yielded the best results from our testing.

The model with eight units performed the best out of the six LSTM models we evaluated with varying units in stacking recurrent layers (8, 16, and 32). One well-known design for effectively managing time series data is LSTMs. In order to increase accuracy and address the forgetting issue, we also experimented with bidirectional data presentation. Recurrent dropout was employed to avoid overfitting. All of these LSTM models showed comparable MAE values, which were continuously lower than the commonsense model.
Lastly, we attempted to combine a 1D convolution model with an RNN.

The hybrid model's higher mean absolute error (MAE) of 3.95 was caused by the convolution's limitations in maintaining the information order. My research indicates that since basic RNNs struggle with the vanishing gradient problem and cannot reliably capture long-term relationships, they should be avoided for time series analysis. Instead, consider more advanced RNN architectures like LSTM and GRU that are designed to overcome these challenges. Our tests show that LSTM is a popular choice for processing time series data, even though GRU might produce better results.

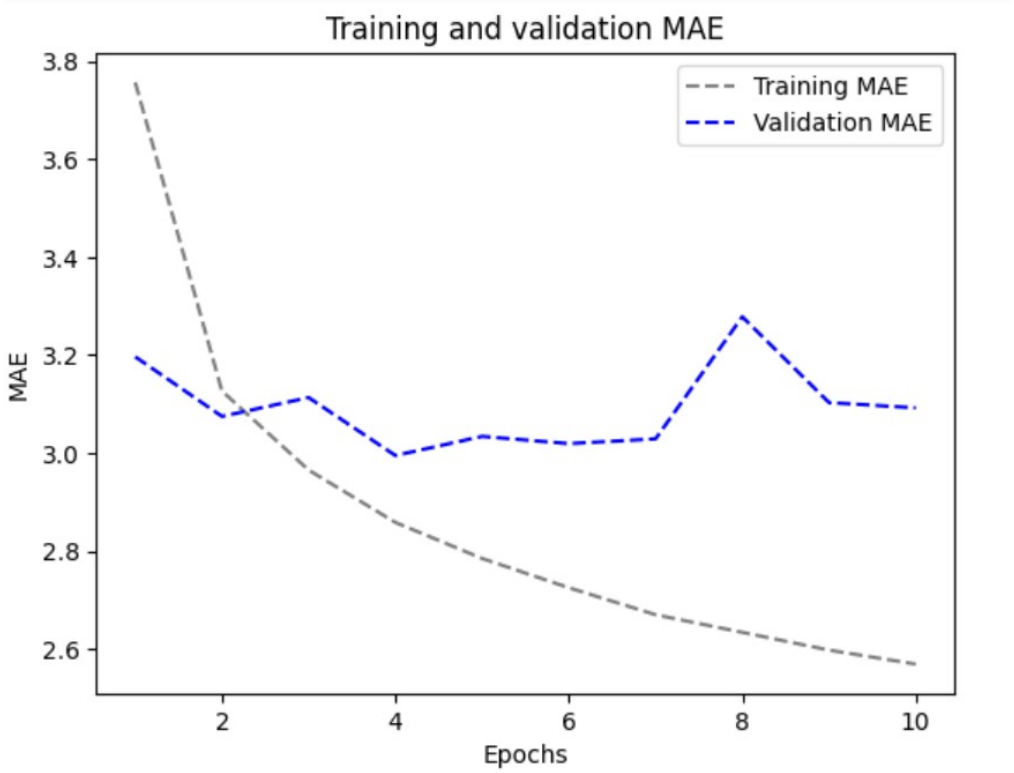
Recurrent dropout rates, the number of units in stacked recurrent layers, and the use of bidirectional data display are examples of hyperparameters that can be adjusted to enhance GRU models. Furthermore, it is recommended to focus on RNN designs made for sequential data because the combination of RNN and 1D convolution did not yield the best results. Convolutional techniques often result in information that is out of order, making them less suitable for processing time series data.

A Basic Machine Learning model:



1D convolutional model:

Given that the input sequences are composed of daily cycles, a convolutional model would be suitable in terms of applying the proper architectural priors. A temporal convolutional network may use the same representations across several days, much as a spatial convolutional network may reuse representations in many places within an image.
This model performs noticeably worse than the densely linked model because not all meteorological data meets the translation invariance criteria; it only achieves a validation MAE of roughly 3.1 degrees, which is far from the tolerable baseline.



A Simple RNN:

Due to its exceptional capacity to incorporate previous time-step information into ongoing decision-making processes, recurrent neural networks (RNNs) are able to identify complex relationships and patterns in sequential data. An RNN's internal state acts as a memory for previous inputs, allowing it to characterize sequences of different lengths.

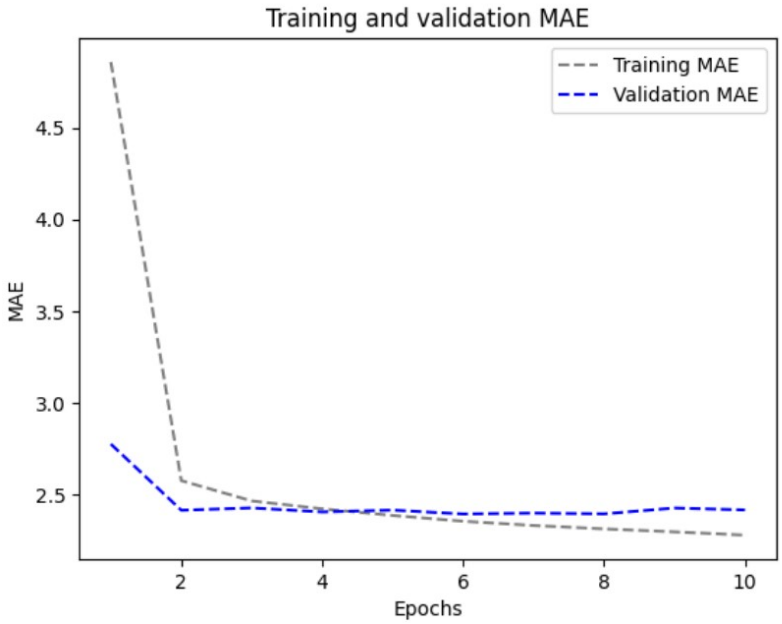
Practical difficulties arise even though a simple RNN may theoretically retain data from all prior times. Deep network training becomes challenging as a result of the vanishing gradient issue. The graph also demonstrates that the simplest RNN performs the worst out of all of them. We must develop LSTM and GRU RNNs as part of Keras in order to solve this issue.

A Simple GRU (Gated Recurrent Unit):

Instead of using LSTM layers, we will employ Gated Recurrent Unit (GRU) layers. Consider GRU as a more straightforward, simplified form of the LSTM architecture, as the two are quite comparable.

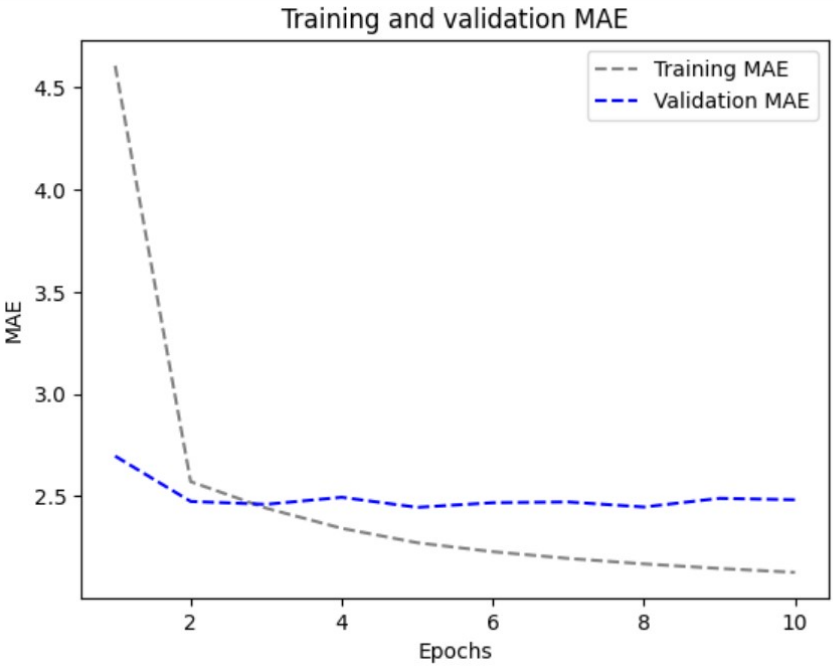
We discovered that the Test MAE of 2.51 is the most effective model when compared to the

others. It correctly captures long-range dependencies in sequential data and uses less processing resources than Long Short-Term Memory (LSTM) models.



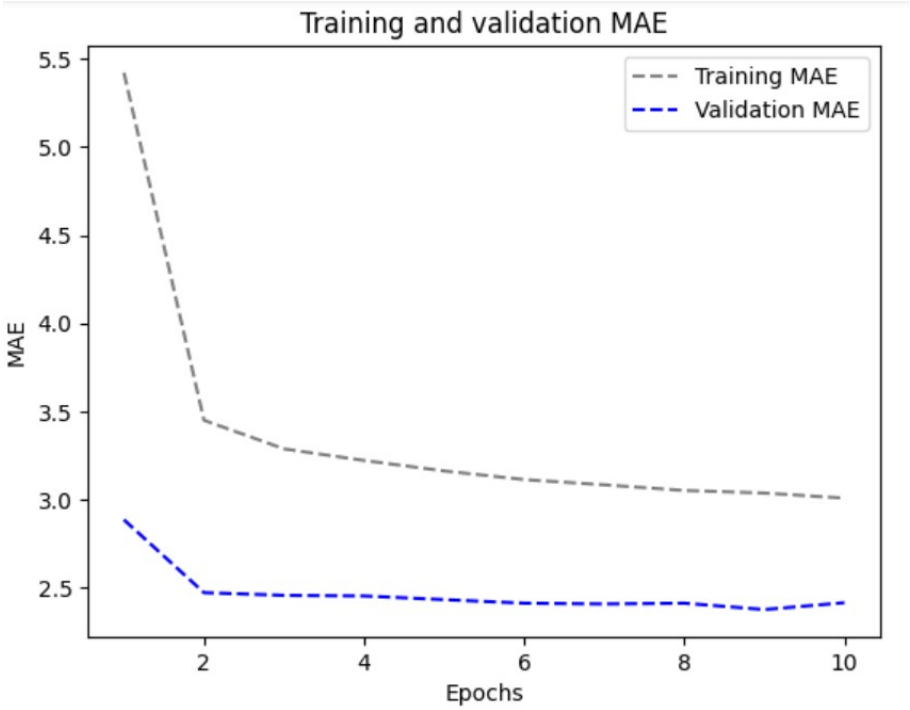
LSTM-Simple:

For this specific use case, recurrent neural networks offer a class of neural network topologies. A highly favored layer among them is the Long Short-Term Memory (LSTM) layer. In a minute, we will see how these models perform by first testing the LSTM layer.



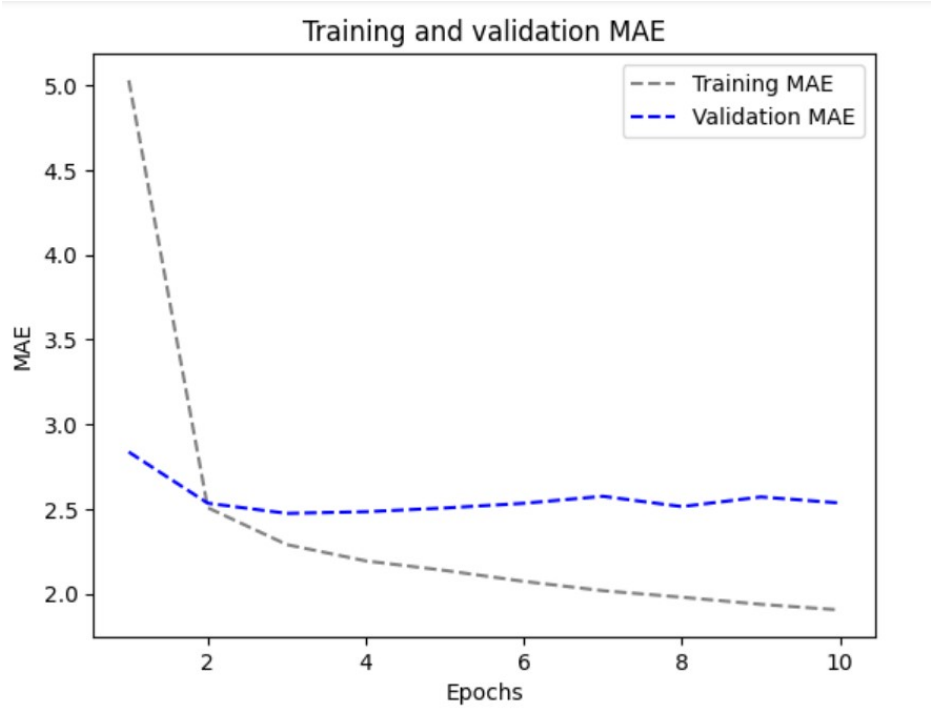
We discover that the test MAE is 2.59 degrees, whereas the validation MAE is as low as 2.59 degrees. Lastly, the LSTM-based model outperforms the commonsense baseline, demonstrating the effectiveness of machine learning in this endeavor (although only somewhat, for the time being).

LSTM - dropout Regularization:

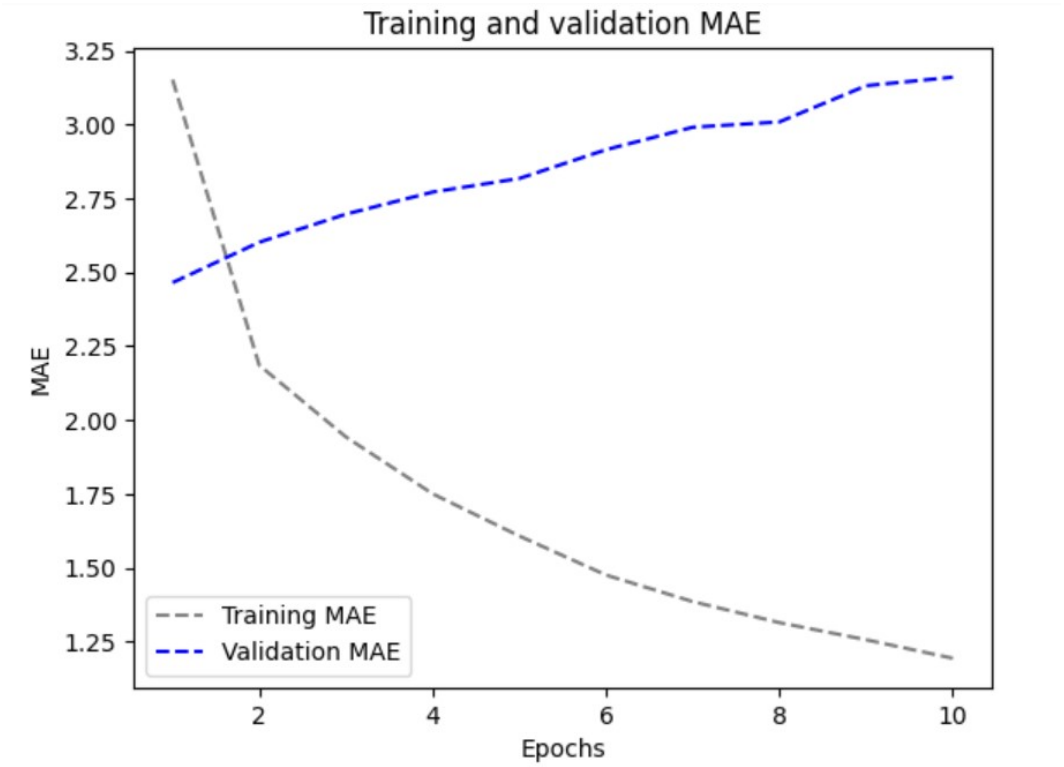


Achievement! There is no longer any overfitting, unlike the first five epochs. We obtain a test MAE of 2.58 degrees and a validation MAE as low as 2.54 degrees.
Not too awful.
Using 8, 16, and 32 units as the various numbers of units inside the stacked recurrent layers, I constructed six distinct LSTM models.

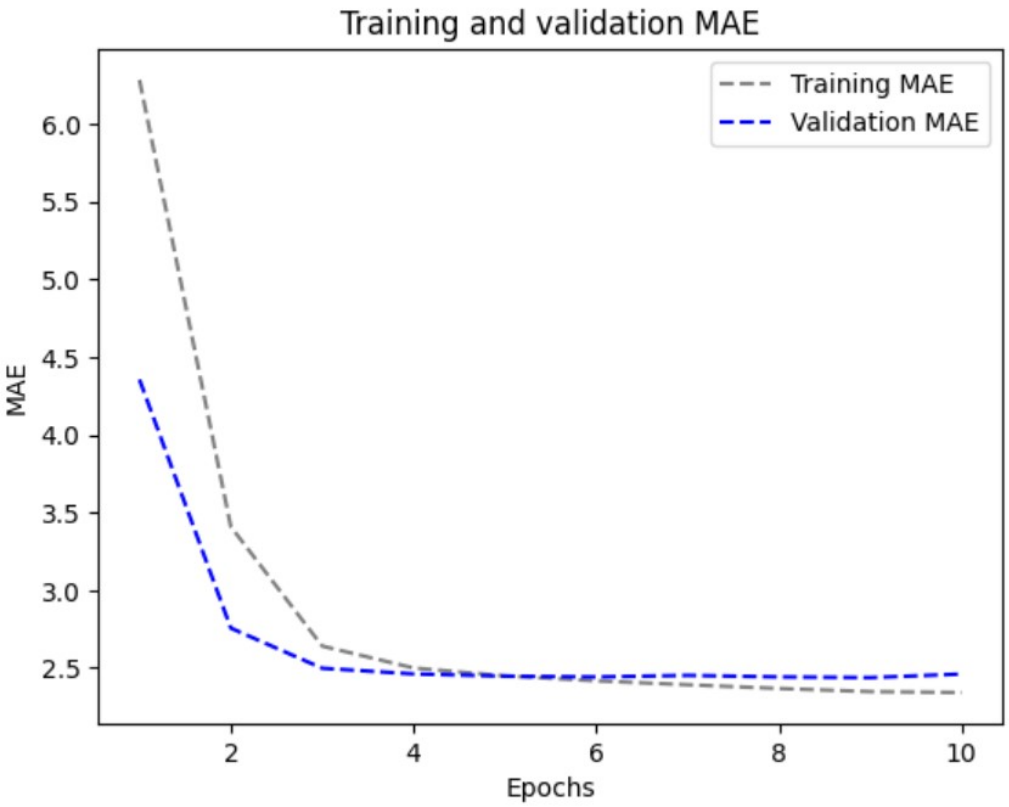
3.LSTM Stacked setup with 16 units:



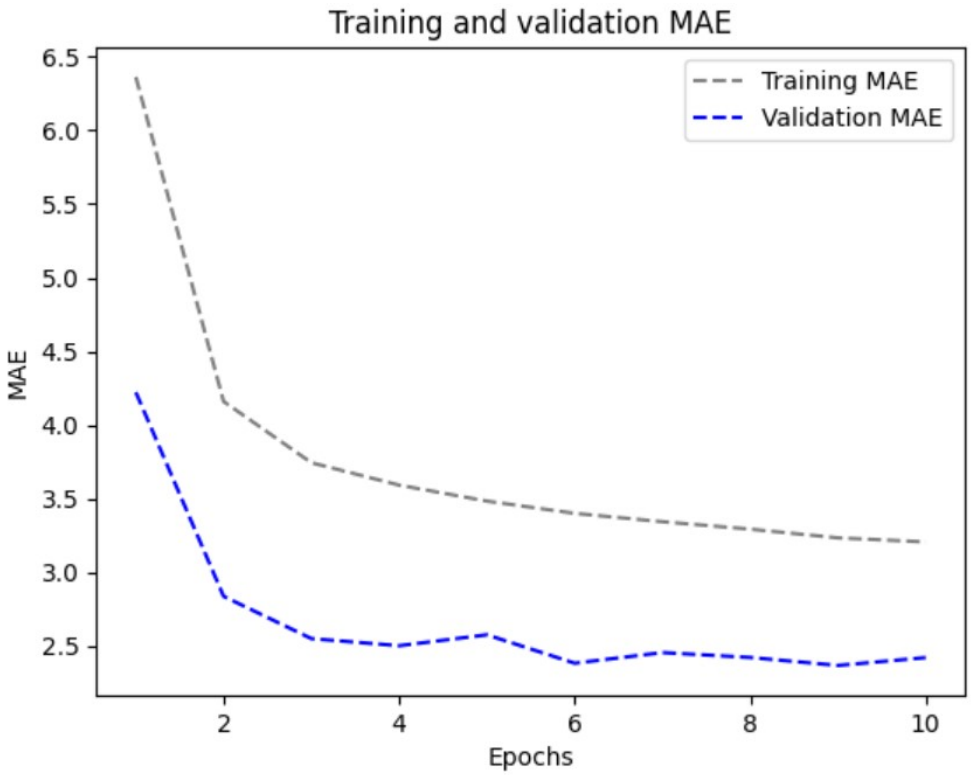
4.LSTM - Stacked setup with 32 units



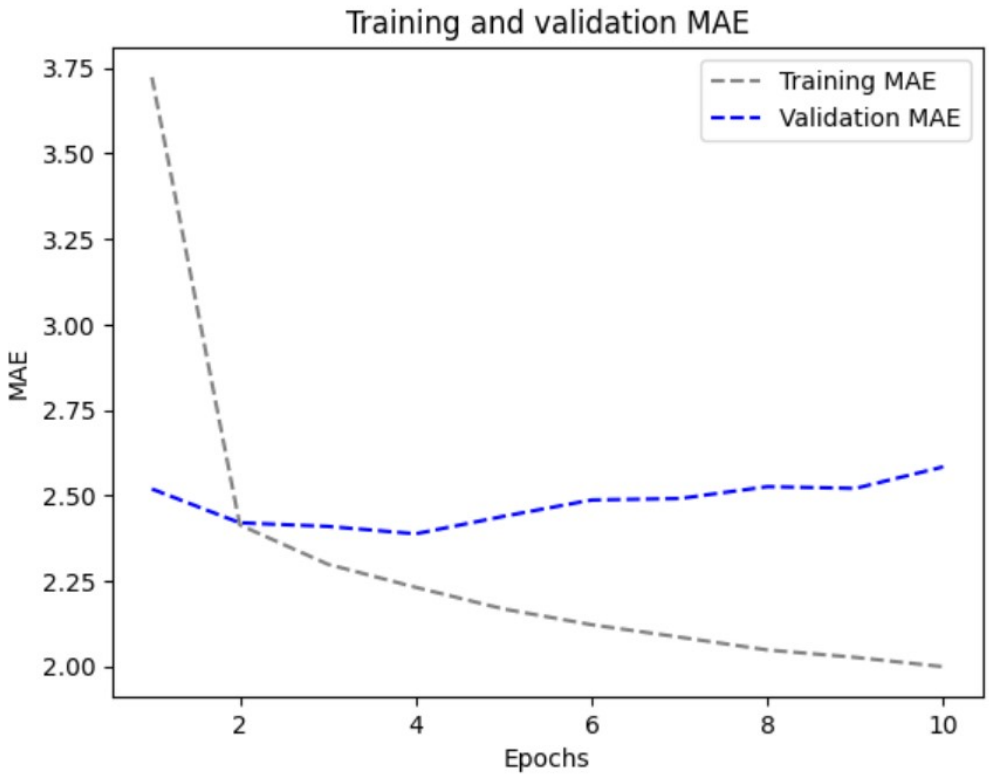
5.LSTM - Stacked setup with 8 units



6.LSTM - dropout-regularized, stacked model

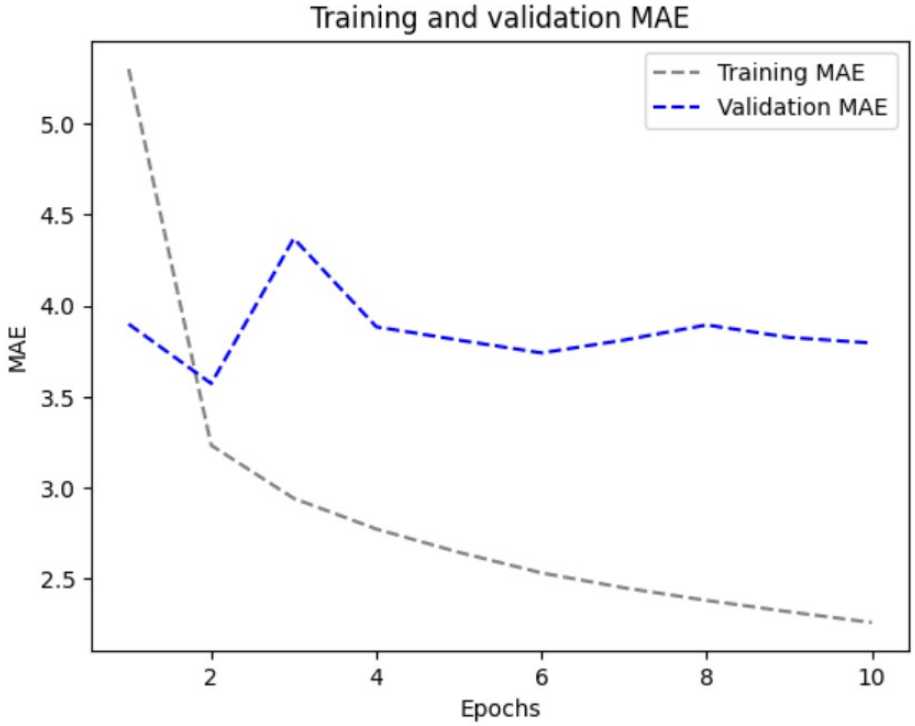


Bidirectional LSTM:

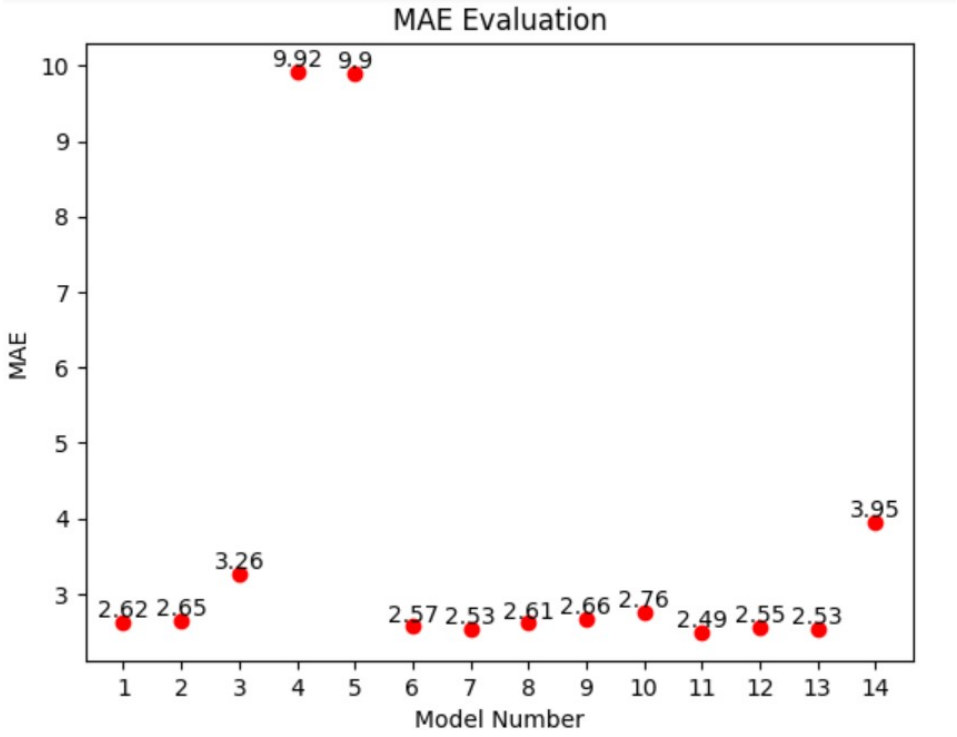


1D Convnets and LSTM togther

With 3.82 MAE, the model I created with RNN and 1D convolution yielded insufficient results. This poor performance could be the result of the convolution limit destroying the information order.



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