TIME SERIES SUMMARY

Akshaya Mamidipalli

811292765

We will illustrate the main differences between timeseries data and the other dataset types we have previously worked with using this temperature forecasting exercise. We shall see that recurrent neural networks (RNNs), a novel machine learning technique, excel at resolving this type of problem, whereas convolutional and highly connected networks are inadequate to handle this type of dataset.

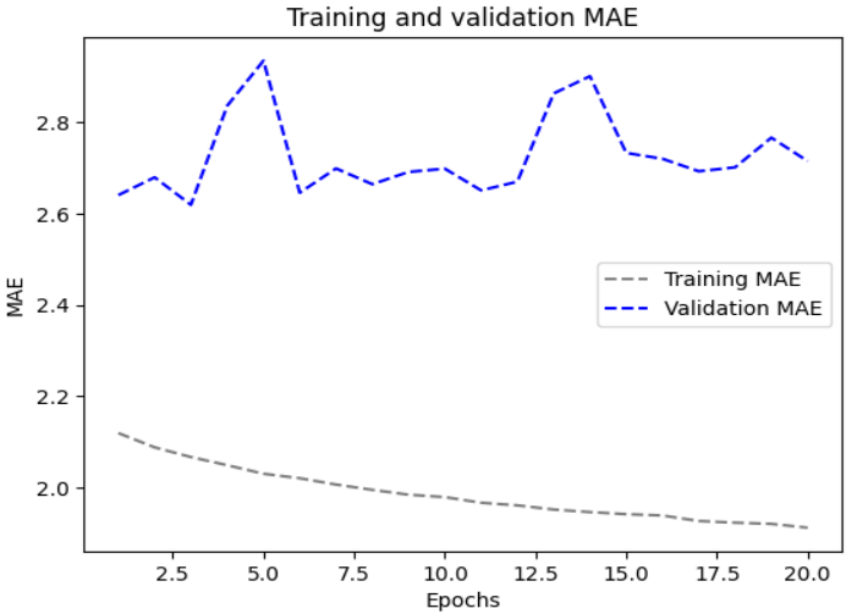
In all our investigations, training will utilize half of the data, validation will use the other quarter, and testing will use the remaining quarter. It is essential to employ validation and test data that are more recent than the training data when working with timeseries data because our objective is to forecast the future based on the past rather than the opposite. This should show up in the validation/test splits. The problem statement will be just as follows: Is it feasible to estimate the temperature in a day using data that has been taken once every hour for the previous five days?

To begin, let's preprocess the data so that it may be used by a neural network. It's easy enough—since the data is already numerical, there is no need to vectorize it.

In all, we built 14 models to analyze time series data. The first model produced a baseline Mean Absolute Error (MAE) of 2.44 using common sense techniques. A slightly higher MAE of 2.62 was obtained when we developed a simple machine learning model with a thick layer. Because the time series data was flattened and the temporal context was lost, the thick layer model performed poorly. Although not always, certain validation losses fall within the range of the no-learning baseline.

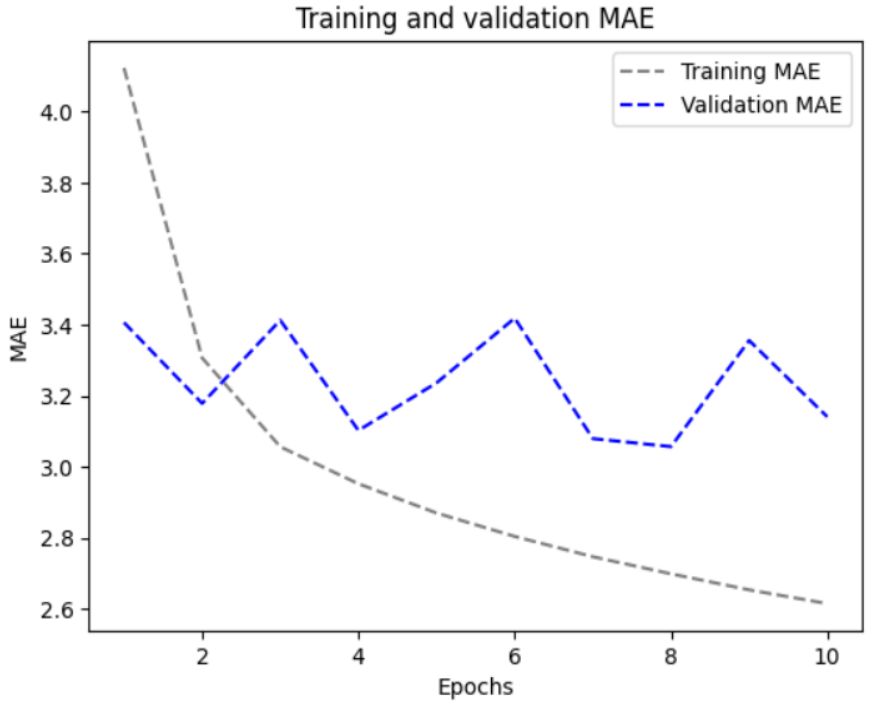
Models :

Basic Dense layer



1D convolutional model

Regarding the use of proper architectural priors, a convolutional model would be appropriate because the input sequences are composed of daily cycles. Similar to how a spatial convolutional network may utilize the same representations in different locations within an image, a temporal convolutional network might do the same throughout many days.
In fact, this model performs much worse than the densely linked model; because not all meteorological data satisfies the translation invariance assumption, it only accomplishes a validation MAE of about 2.9 degrees, which is far from the realistic baseline.



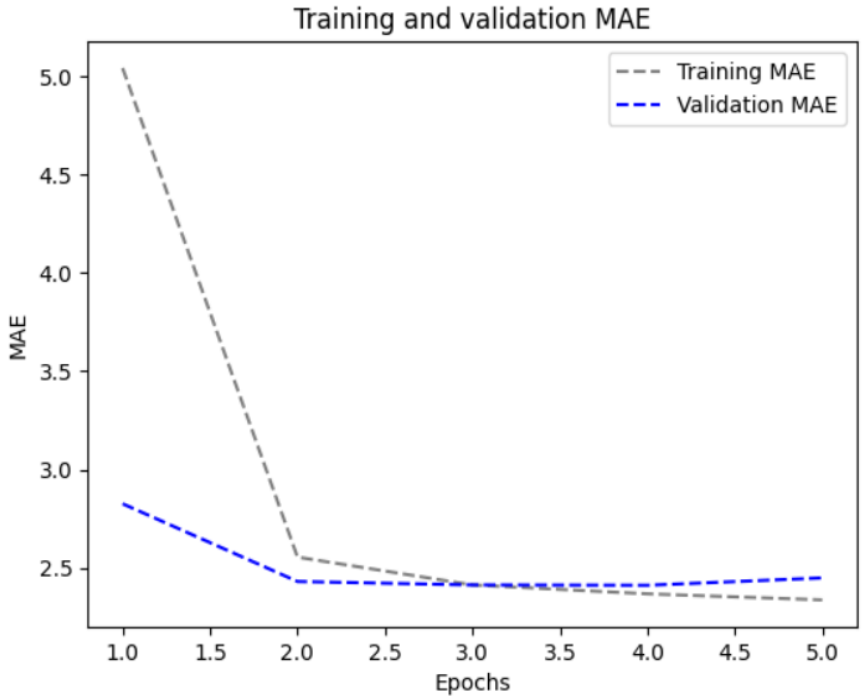
A Simple RNN

Recurrent neural networks (RNNs) are able to identify intricate relationships and patterns in sequential data because of their remarkable ability to integrate prior time step information into current decision-making processes. Since the internal state of an RNN serves as a memory of prior inputs, sequences of varying durations can be described. Although a basic RNN may theoretically maintain data from all previous times, there are practical challenges. The vanishing gradient issue makes deep network training difficult as a result. Additionally, the graph shows that, of all the RNNs, the simplest one performs the worst.

To overcome this problem, as a part of Keras, we have to create LSTM and GRU RNNs.

GRU

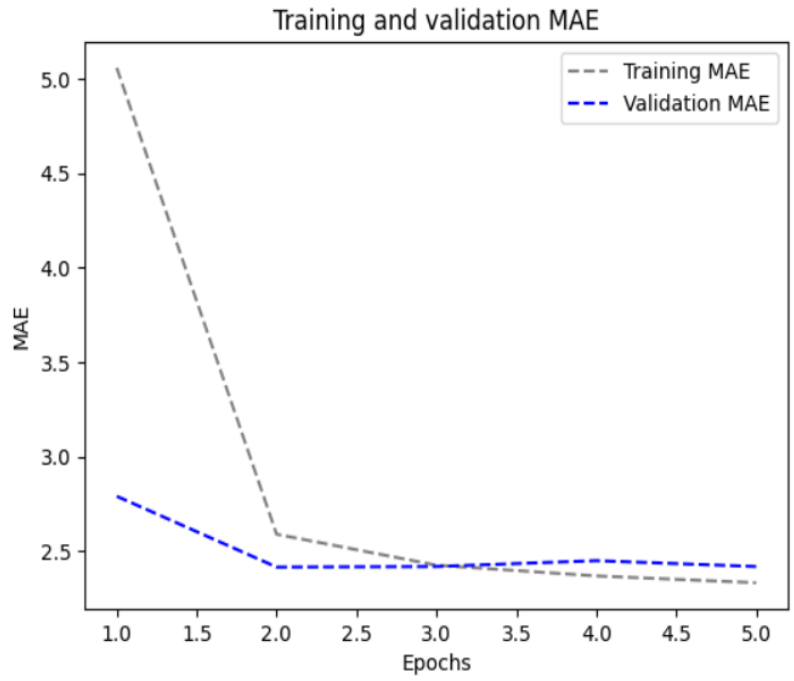
We will use Gated Recurrent Unit (GRU) layers in place of LSTM layers. GRU and LSTM are somewhat comparable; think of it as a condensed, simpler version of the LSTM architecture.



In comparison to the other models, we found that Test MAE-2.54 is the most efficient model, has lower computational costs than Long Short-Term Memory (LSTM) models, and successfully captures long-range dependencies in sequential data.

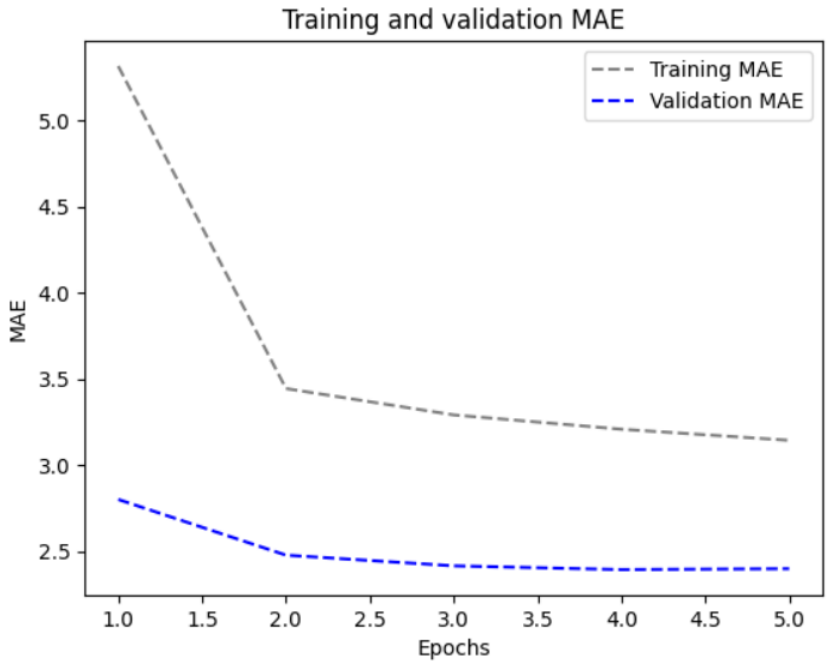
LSTM

A family of neural network topologies specifically designed for this use case is offered by recurrent neural networks. The Long Short-Term Memory (LSTM) layer is among the most popular of them. In a minute, we will examine how these models work after testing the LSTM layer.



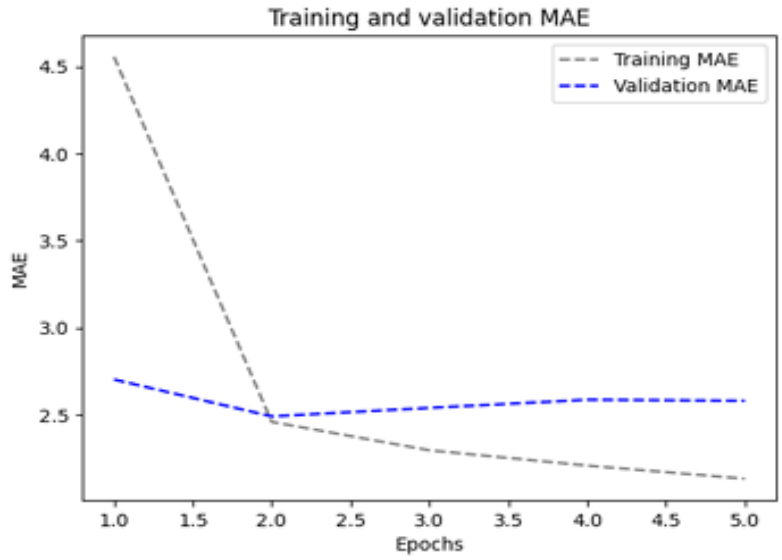
Much better! We get a validation MAE as low as 2.39 degrees and a test MAE of 2.57 degrees. Lastly, the LSTM-based model shows how effective machine learning is in this quest by outperforming the commonsense baseline (although only somewhat, at least for the time being).

LSTM - dropout Regularization

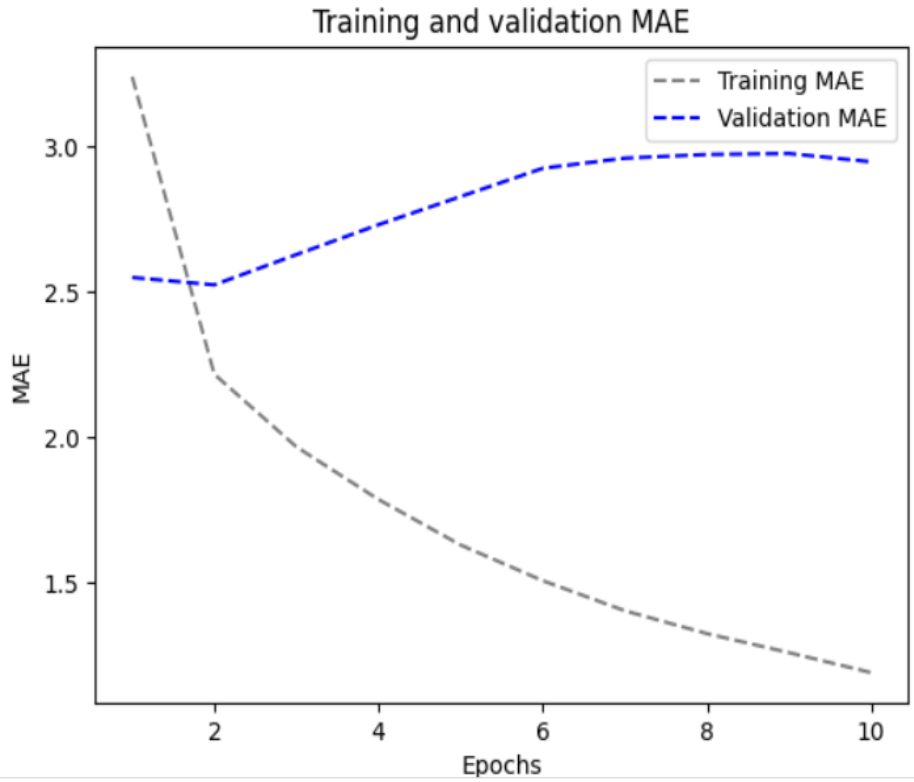


Success! The overfitting that was present throughout the initial five epochs has ceased. We achieve a validation MAE as low as 2.36 degrees and a test MAE of 2.56 degrees. Not too bad.
I created six different LSTM models using 8, 16, and 32 units as different numbers of units inside the stacked recurrent layers.

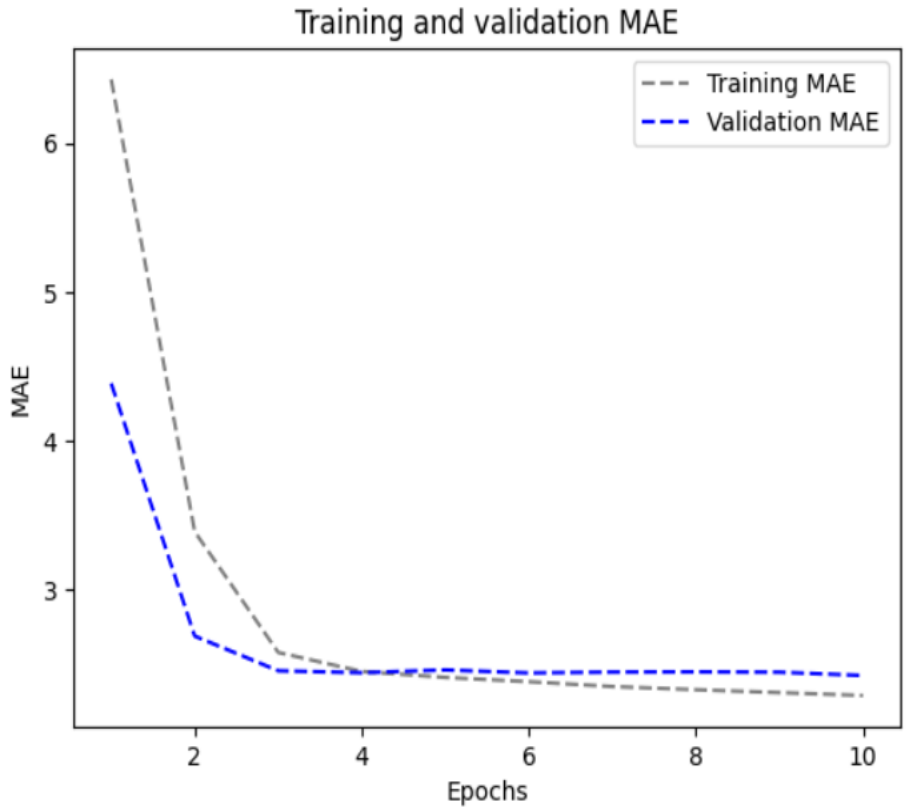
LSTM - Stacked setup with 16 units



LSTM - Stacked setup with 32 units

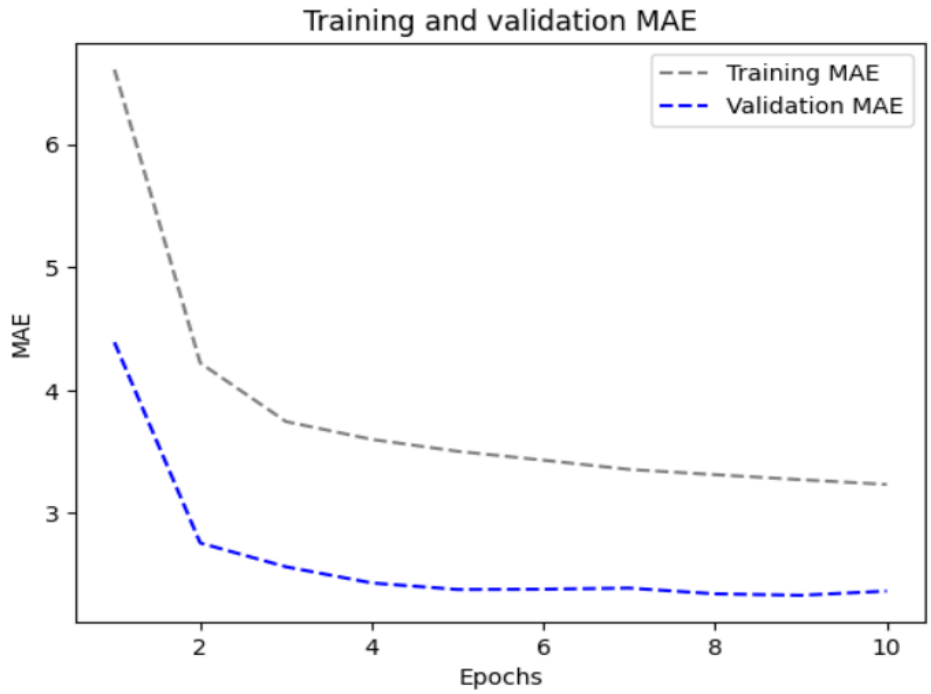


LSTM - Stacked setup with 8 units

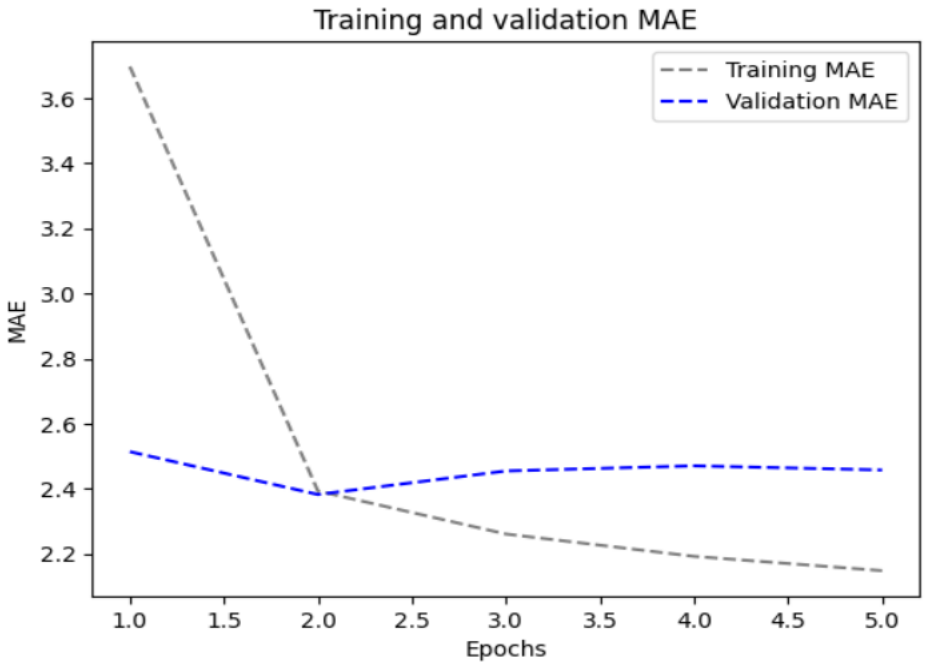


The 8-unit arrangement performed the best out of these options, with an MAE score of 2.58. I also experimented with techniques like bidirectional data, which reduces MAE values by providing information to a recurrent network in many directions, and recurrent dropout, which counteracts overfitting, to improve the model. All of the models' MAE values were similar as a result of these enhancements, but most astonishingly, they were constantly lower than those of the commonsense model, which is also supported by the MAE assessment graph.

LSTM - dropout-regularized, stacked model

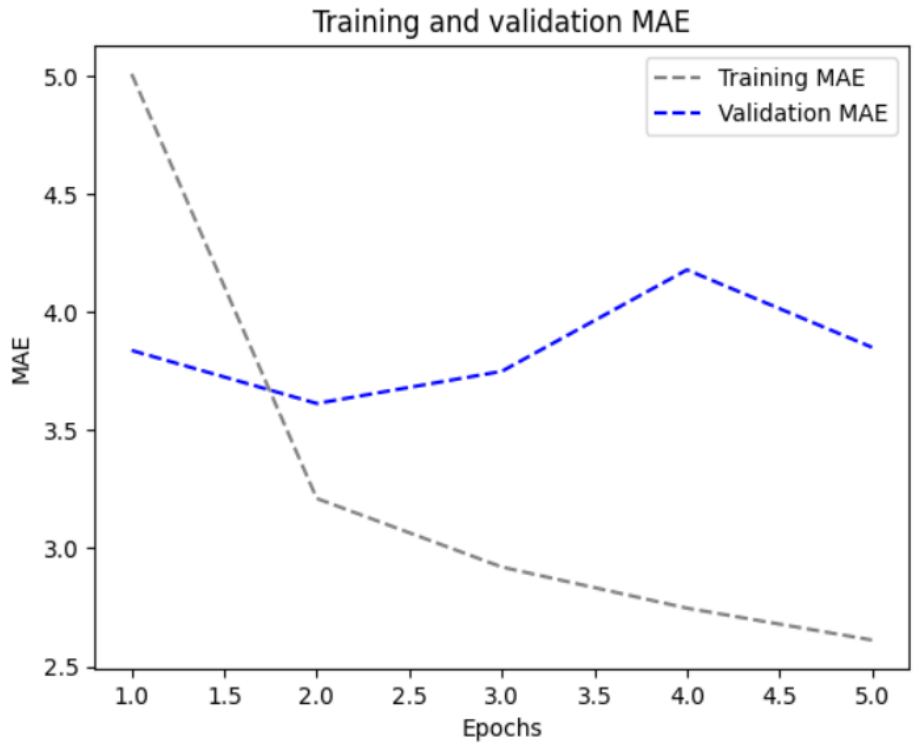


Bidirectional LSTM

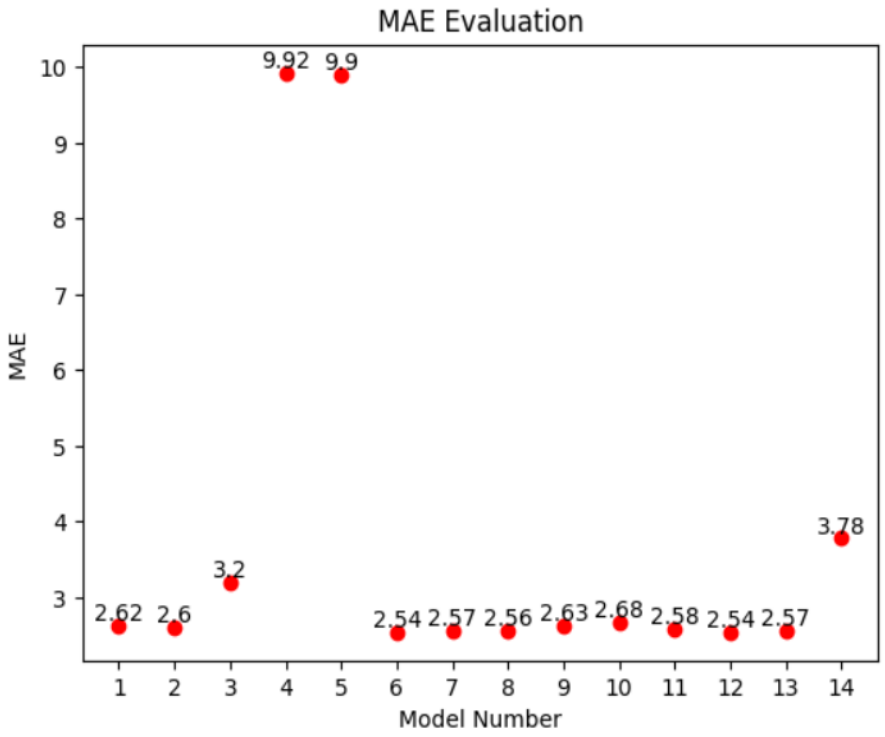


1D Convnets and LSTM together

The model, which I developed using both RNN and 1D convolution, produced inadequate outcomes with 3.78 MAE. The information order may be being destroyed by the convolution limit, which might be the cause of this subpar performance.



All Models Performance



Results

MODEL	Validation MAE	Test MAE
Dense model	2.66	2.60
1D convolutional Model	2.98	3.2
Simple RNN	9.83	9.92
Stacked Simple RNN	9.80	9.90
GRU	2.43	2.54
LSTM simple	2.39	2.57
LSTM -dropout Regularization	2.36	2.56
LSTM- Stacked 16 units	2.58	2.63
LSTM – Stacked 32 units	2.94	2.68
LSTM – Stacked 8 units	2.42	2.58
LSTM – dropout Regularization, stacked model	2.36	2.57
Bidirectional LSTM	2.45	2.57
1D convolutional and LSTM	3.84	3.78

In conclusion, my findings indicate that using LSTM and GRU (advanced RNN designs) is preferable than combining RNN with 1D convolution, which yielded subpar outcomes. Bidirectional LSTM is still a common option for handling time series data, however after some experimentation, I think GRU is a better option. The number of units in the stacked recurrent layers, the recurrent dropout rate, and the utilization of bidirectional data are hyperparameters that need to be changed in order to optimize GRU.