ASSIGNMENT-3
WEATHER TIME SERIES FORCASTING
SUMMARY

Dinesh Yechuri
KSU ID : 811295667

We will illustrate the main differences between timeseries data and the other dataset types we've already worked with using this temperature-forecasting exercise. We will see that recurrent neural networks (RNNs), a novel type of machine learning technique, totally flourish at addressing this kind of problem, whereas convolutional and highly linked networks are unable to handle this kind of dataset.

In all our research, half of the data will be used for training, twenty-five percent for validation, and the remaining twenty-five percent for testing. When working with timeseries data, it is essential to use validation and test data that is more recent than the training data because our objective is to predict the future based on the past rather than the opposite. This should be reflected in the validation/test splits.
The issue will be precisely formulated as follows: Is it possible to predict the temperature of a given day using information gathered every hour over the previous five days?

To enable a neural network to learn how to use the input, let's begin by preprocessing it. Easy enough, since the data is already numerical and we don't need to conduct any vectorization.

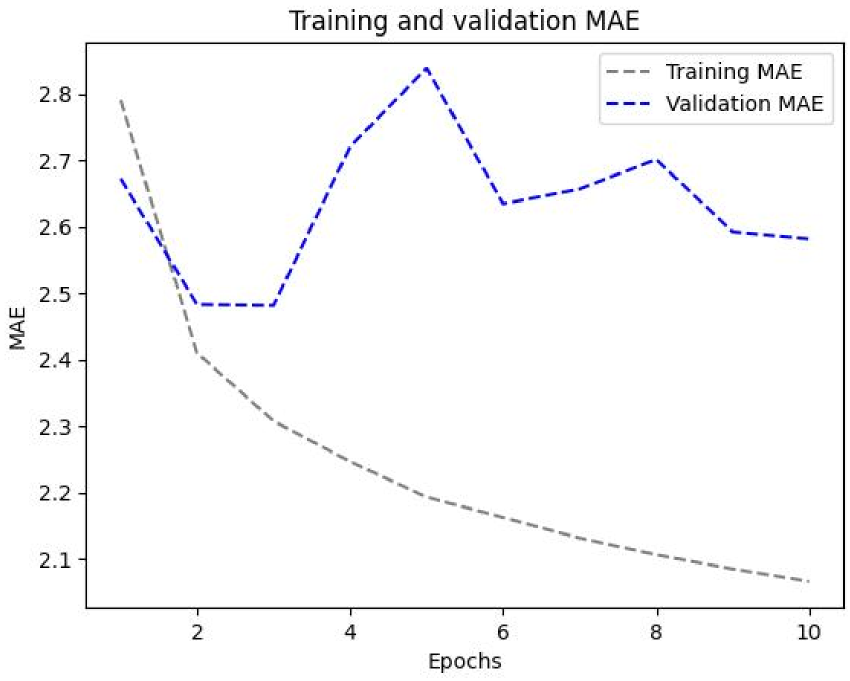
In order to analyze time series data, we created 14 different models. Using common-sense techniques, the first model provided a baseline and produced a Mean Absolute Error (MAE) of 2.62. After that, we developed a simple machine learning model with a dense layer, which produced an MAE of 2.60 that was marginally higher. The flattening of the time series data, which eliminated the temporal context, resulted in poor performance of the dense layer model. Additionally, a convolutional model was used, but it produced subpar results since it treated every data segment equally—even after pooling—disturbing the sequential order of the data.

Recurrent Neural Networks (RNNs) are therefore more appropriate for time series data, as we have discovered. The ability of Recurrent Neural Networks (RNNs) to integrate information from previous steps into their current decision-making process is a crucial characteristic. As a result, the network can find patterns and dependencies in sequential data. The internal state of the RNN functions as a kind of memory, storing data from previous inputs and enabling it to simulate sequences of different lengths. Nevertheless, the fundamental Simple RNN is frequently overly straightforward to be truly useful. Interestingly, Simple RNN has a major flaw: as the graphical representation shows, it consistently performs the worst out of all the models. Although Simple RNN should theoretically be able to preserve data from every prior time step, it typically practically, particularly in deep networks where the infamous "vanishing gradient problem" exists. Because of this issue, the network is essentially untrainable. More sophisticated RNN variations, including the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), were created in response to this difficulty and are incorporated into Keras. Because the simple GRU model is more computationally efficient than LSTMs and can capture long-range dependencies in sequential data, our experimentation with it produced the best results of all the models.

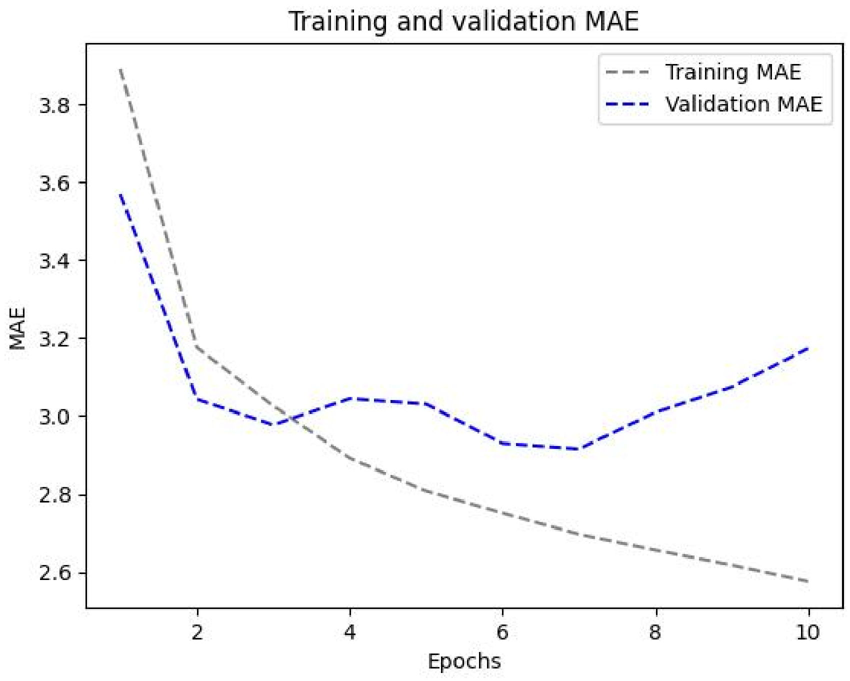
We tested six different LSTM models with different units in stacking recurrent layers (8, 16, and 32), and the model with 8 units showed the greatest performance. LSTMs are a well-known architecture for handling time series data successfully. We also experimented with bidirectional data presentation to improve accuracy and solve the forgetting problem, and we used recurrent dropout to prevent overfitting. Similar MAE values, which were consistently lower than the common-sense model, were displayed by all of these LSTM models.

Finally, we tried to integrate an RNN with a 1D convolution model. The hybrid model produced a greater mean absolute error (MAE) of 3.76, 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 architectures 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. Consider adjusting hyperparameters like the number of units in stacked recurrent layers, recurrent dropout rates, and other factors to enhance GRU models as well as the application of bidirectional data display. Additionally, since the combination of RNN with 1D convolution did not produce 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.

A basic machine-learning model



1D convolutional model



1D convolutional architecture :

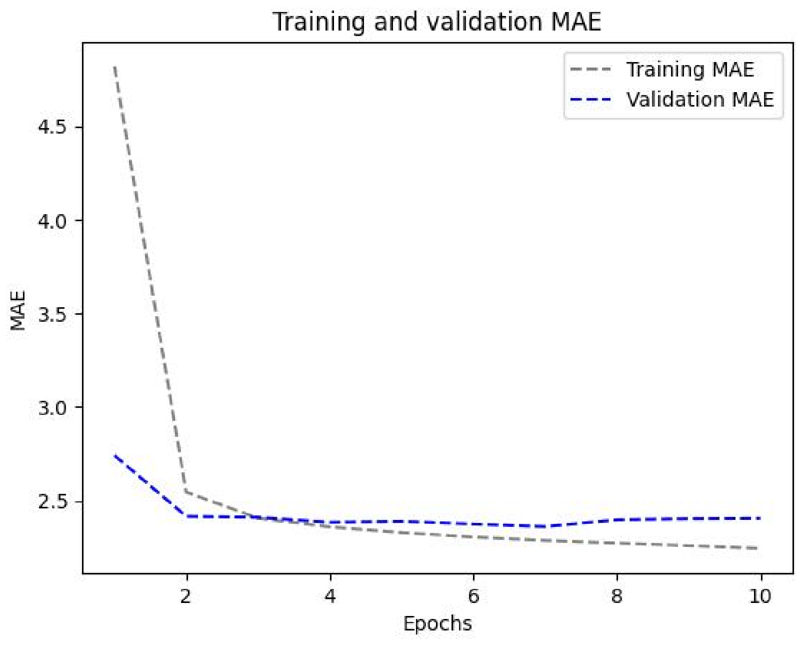
A convolutional model would be adequate in terms of using appropriate architectural priors, considering that the input sequences are made up of daily cycles. Given that a spatial convolutional network could just as a temporal convolutional network may utilize the same representations across several days, it may also reuse the same representations across several locations in an image.

Because not all meteorological data satisfies the translation invariance condition, this model performs significantly worse than the densely linked model; it only accomplishes a validation MAE of about 3.1 degrees, which is far from the reasonable baseline.

A Simple RNN :

A basic RNN Recurrent neural networks (RNNs) possess the remarkable ability to integrate past time step data into current decision-making processes, allowing them to identify intricate relationships and patterns in consecutive data. Since an RNN's internal state serves as a memory of prior inputs, it is possible to define sequences of varying duration. Even though a basic RNN may theoretically maintain data from all previous times, practical challenges still occur. This leads to the vanishing gradient problem, which makes training deep networks difficult. Additionally, the graph shows that out of all of them, the simplest RNN performs the worst.

A Simple GRU (Gated Recurrent Unit)



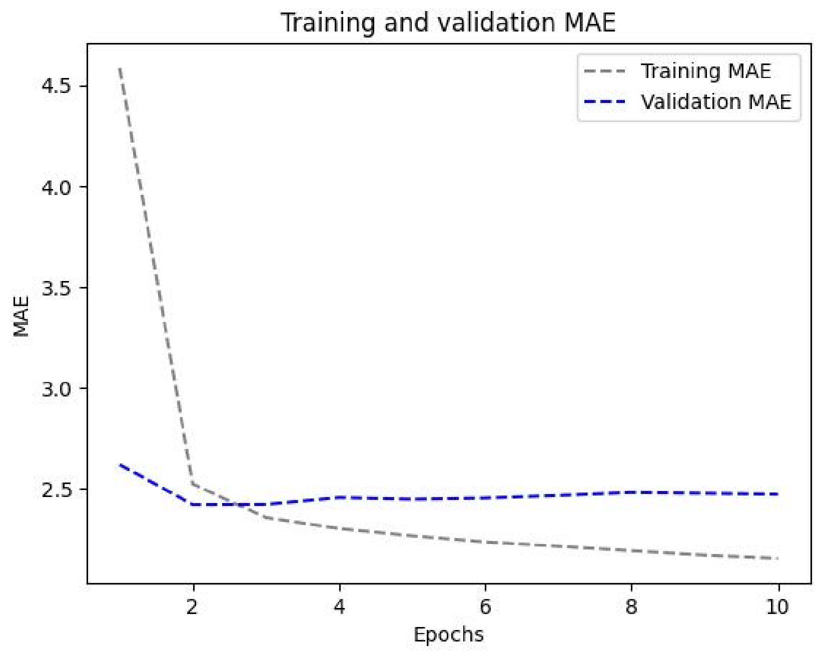
GRU :

We will use Gated Recurrent Unit (GRU) layers in place of LSTM layers. GRU and LSTM are somewhat similar; think of it as a simplified, more basic version of the LSTM architecture.

We found that Test MAE – 2.47 is the most efficient model, requiring lowest processing power.

more expensive than Long Short-Term Memory (LSTM) models, but in comparison to the other models, it captures long-range dependencies in sequential data more effectively.

LSTM-Simple

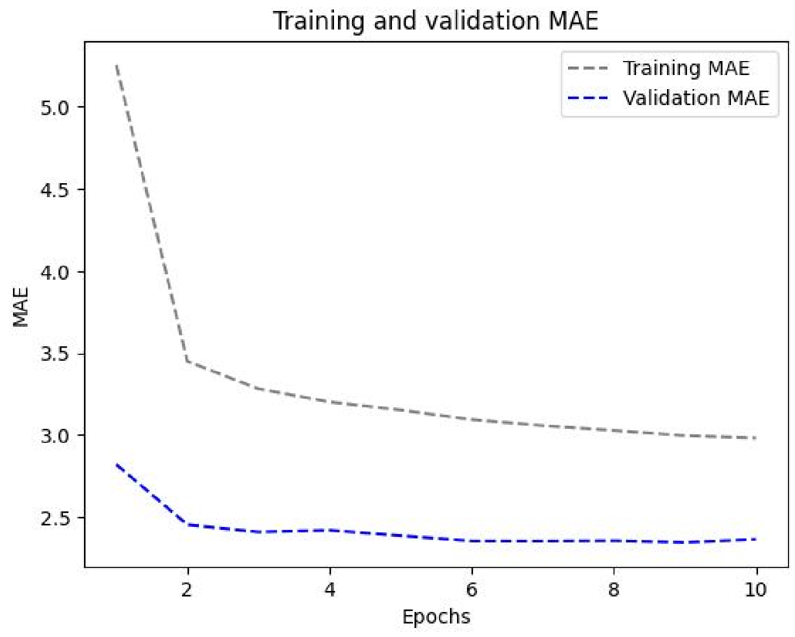


LSTM :

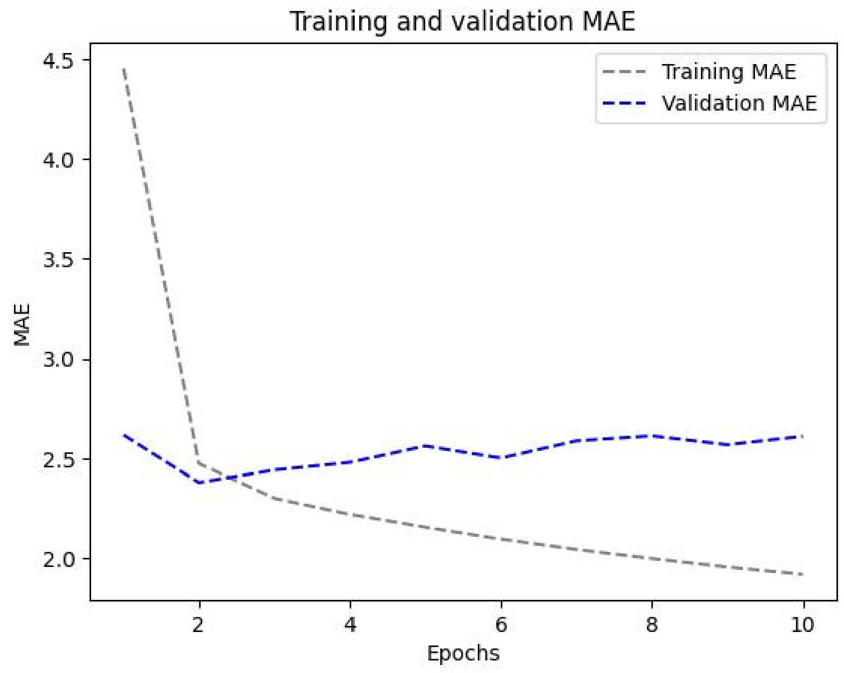
A family of neural network topologies especially designed for this use case is offered by LSTM Recurrent Neural Networks. The Long Short-Term Memory is one of the most popular ones.

layer (LSTM). We'll see how these models work in a moment as we first test the LSTM layer.

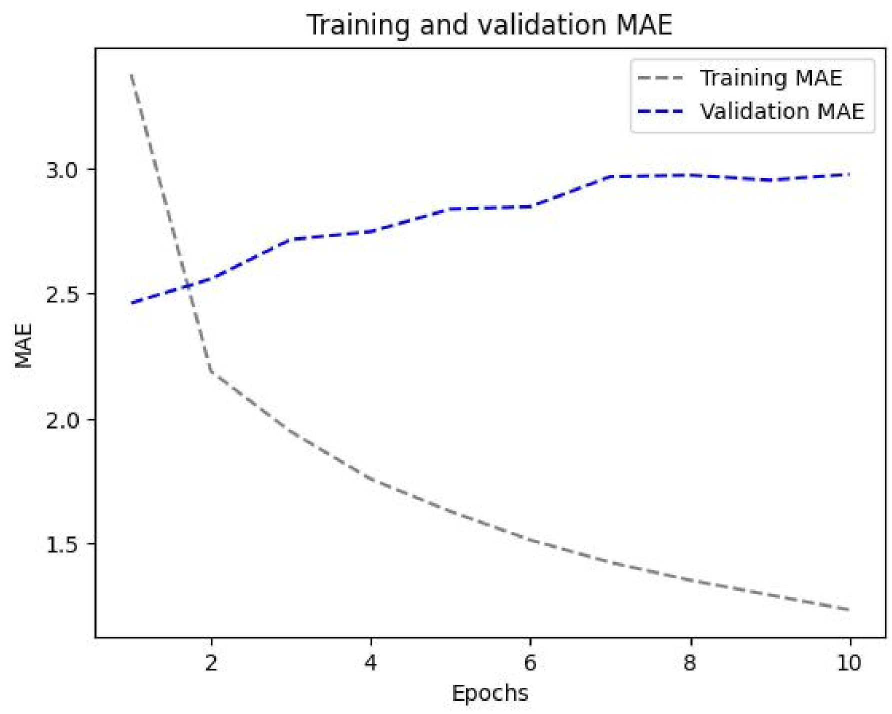
LSTM - dropout Regularization



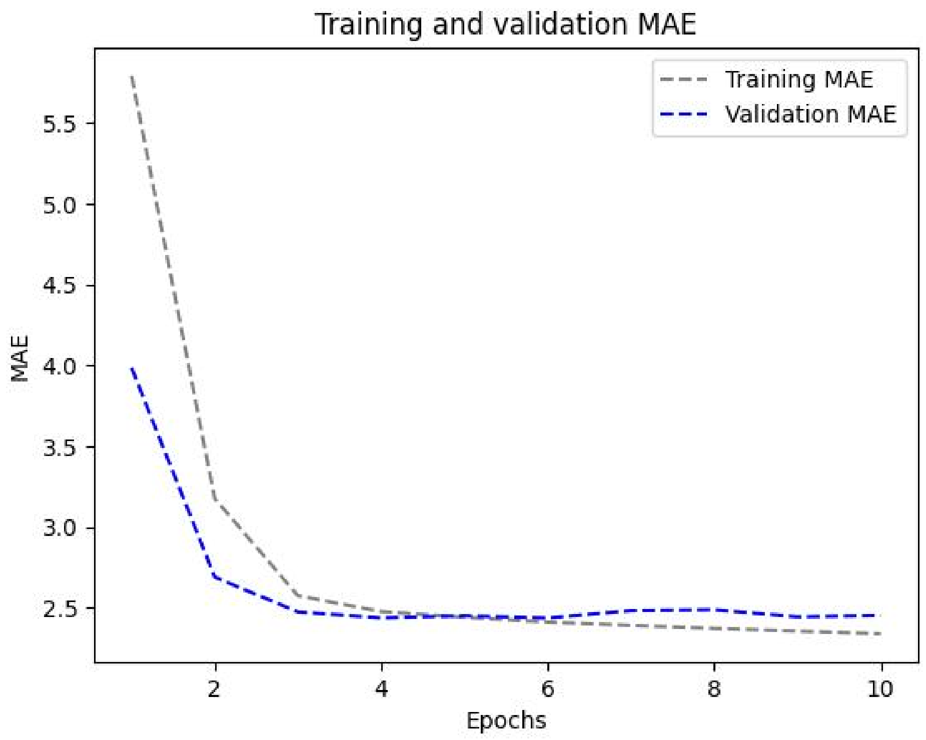
Stacked setup with 16 units



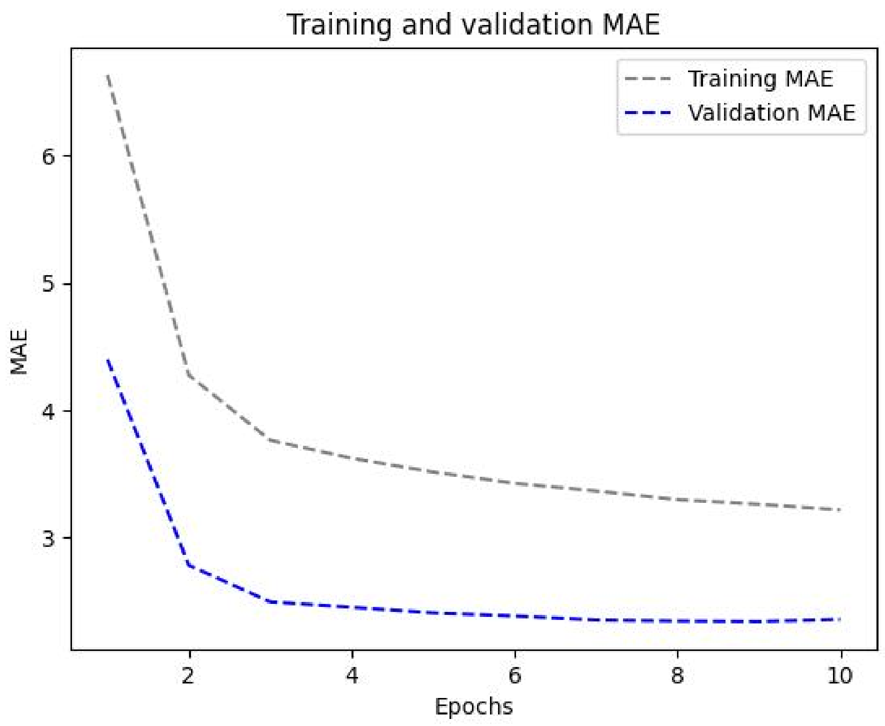
Stacked setup with 32 units



Stacked setup with 8 units

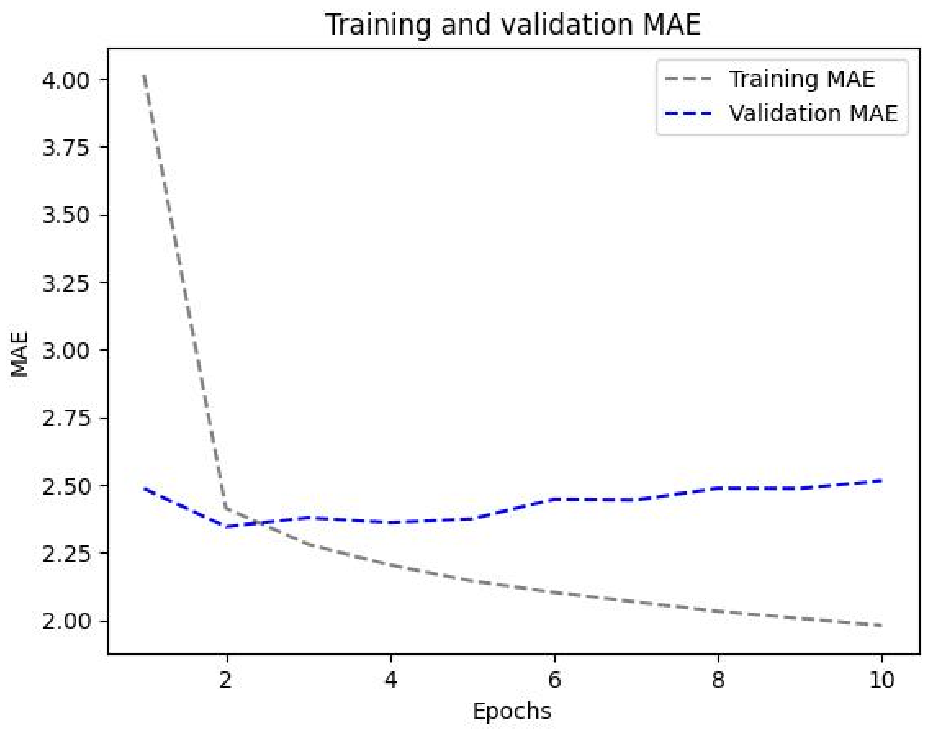


LSTM - dropout-regularized, stacked



Achievement! The overfitting that occurred over the first 5 epochs has stopped. We attain a test MAE of 2.54 degrees and a validation MAE as low as 2.35 degrees. Not too horrible. Using 8, 16, and 32 units as varying numbers of units inside the stacked recurrent layers, I constructed six distinct LSTM models.

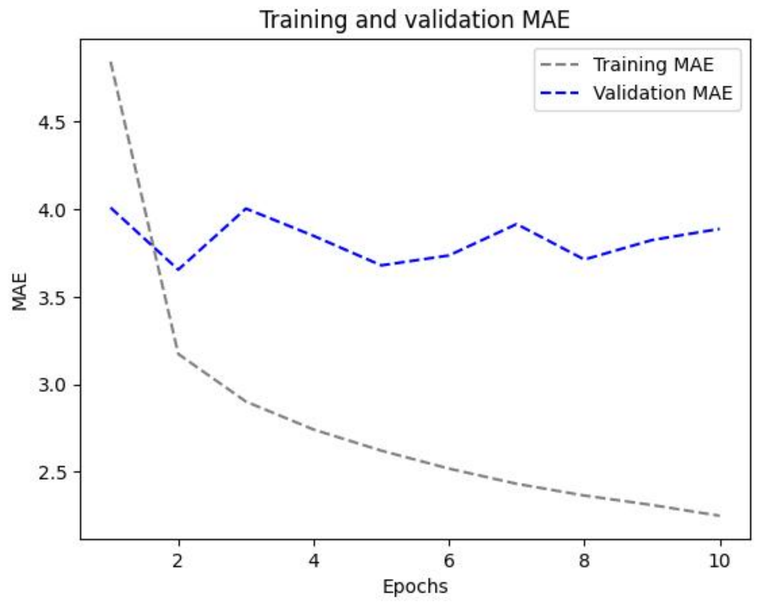
Bidirectional LSTM



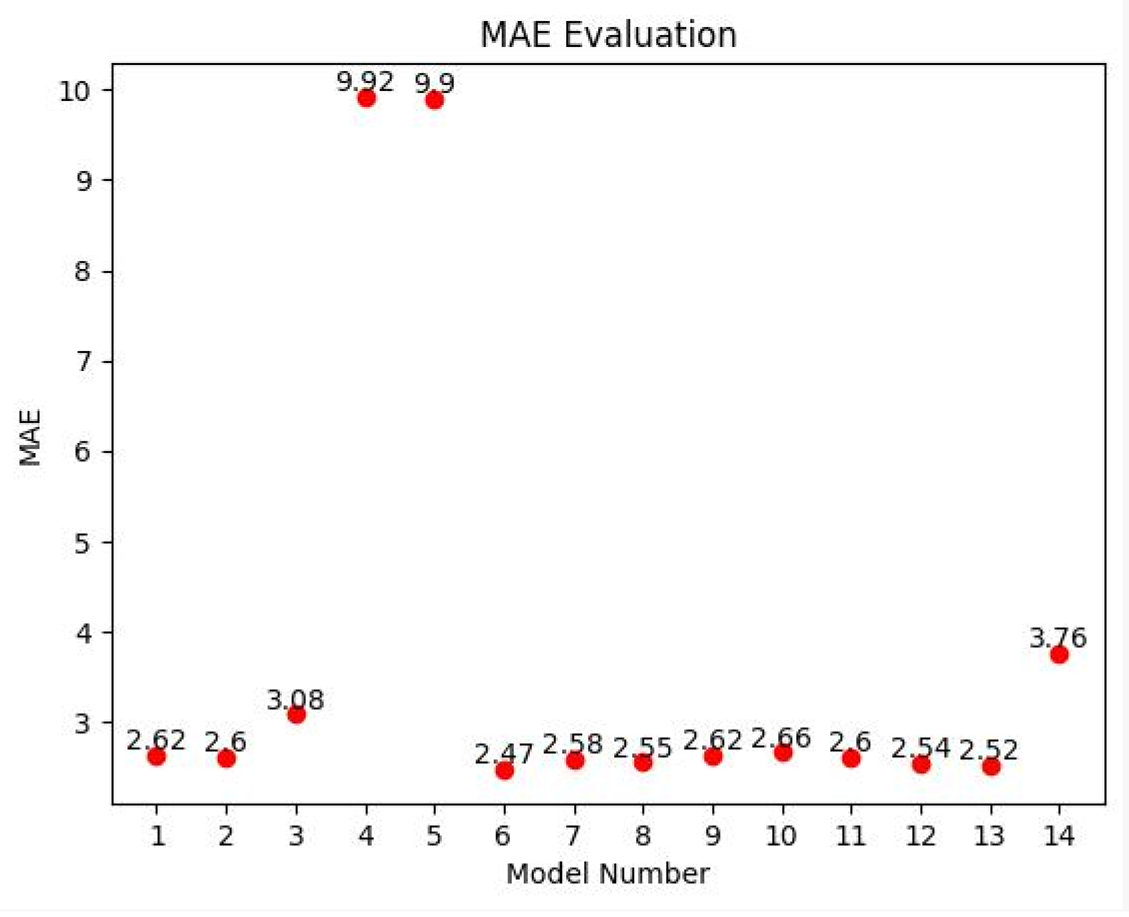
1D Convnets and LSTM together

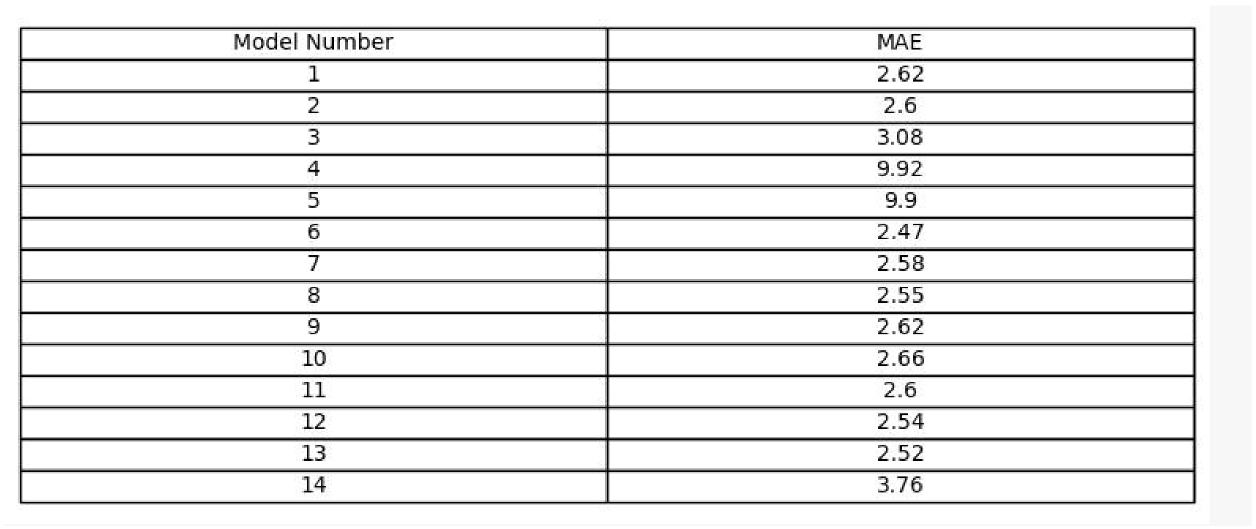
With 3.76 MAE, the model I created with RNN and 1D convolution yielded insufficient results. The convolution may be destroying the information order.

constraint, which could be the reason for this poor performance.



MAE Evaluation:





Model Number	MAE
1	2.62
2	2.6
3	3.08
4	9.92
5	9.9
6	2.47
7	2.58
8	2.55
9	2.62
10	2.66
11	2.6
12	2.54
13	2.52
14	3.76

In conclusion, my findings indicate that using LSTM and GRU (advanced RNN architectures) is the preferable choice, while combining RNN with 1D convolution yielded subpar outcomes. Although bidirectional LSTM is still a popular alternative, I think that GRU is a more efficient method for processing time series data after considerable testing. The number of units in the stacked recurrent layers, the recurrent dropout rate, and the utilization of bidirectional data are examples of hyperparameters that should be changed to optimize GRU.