Assignment - 3

Time-Series Data Report

Sushma Chiluveru

We will use this temperature-forecasting exercise to highlight the key distinctions between time-series data and other types of datasets we've previously worked with. It will be evident that recurrent neural networks (RNNs), a unique form of machine learning, excel at addressing this type of problem, while convolutional and densely connected networks struggle to manage this kind of data.

In all our studies, we will allocate 50% of the data for training, 25% for validation, and the remaining 25% for testing. It is crucial to use more recent data for validation and testing compared to the training data when working with time series data. This is because our goal is to predict future events based on past information, rather than the other way around.

This is what the test/validation splits ought to show. The specific formulation of the problem will be as follows: Can the temperature of a day be predicted using data that has been collected every hour for the past five days?

First, the data is preprocessed to make it suitable for training a neural network. Since the data is already numerical, no vectorization is needed. We created 14 models to analyze the time series data. Using a Mean Absolute Error (MAE) of 2.44 as a baseline, the first model employed common sense techniques. In comparison, our thin-layer machine learning model had a slightly higher MAE of 2.62.

The time series data was flattened, causing the temporal context to be lost, which resulted in poor performance from the thick-layer model. Some of the validation losses are close to the no-learning baseline, but not consistently.

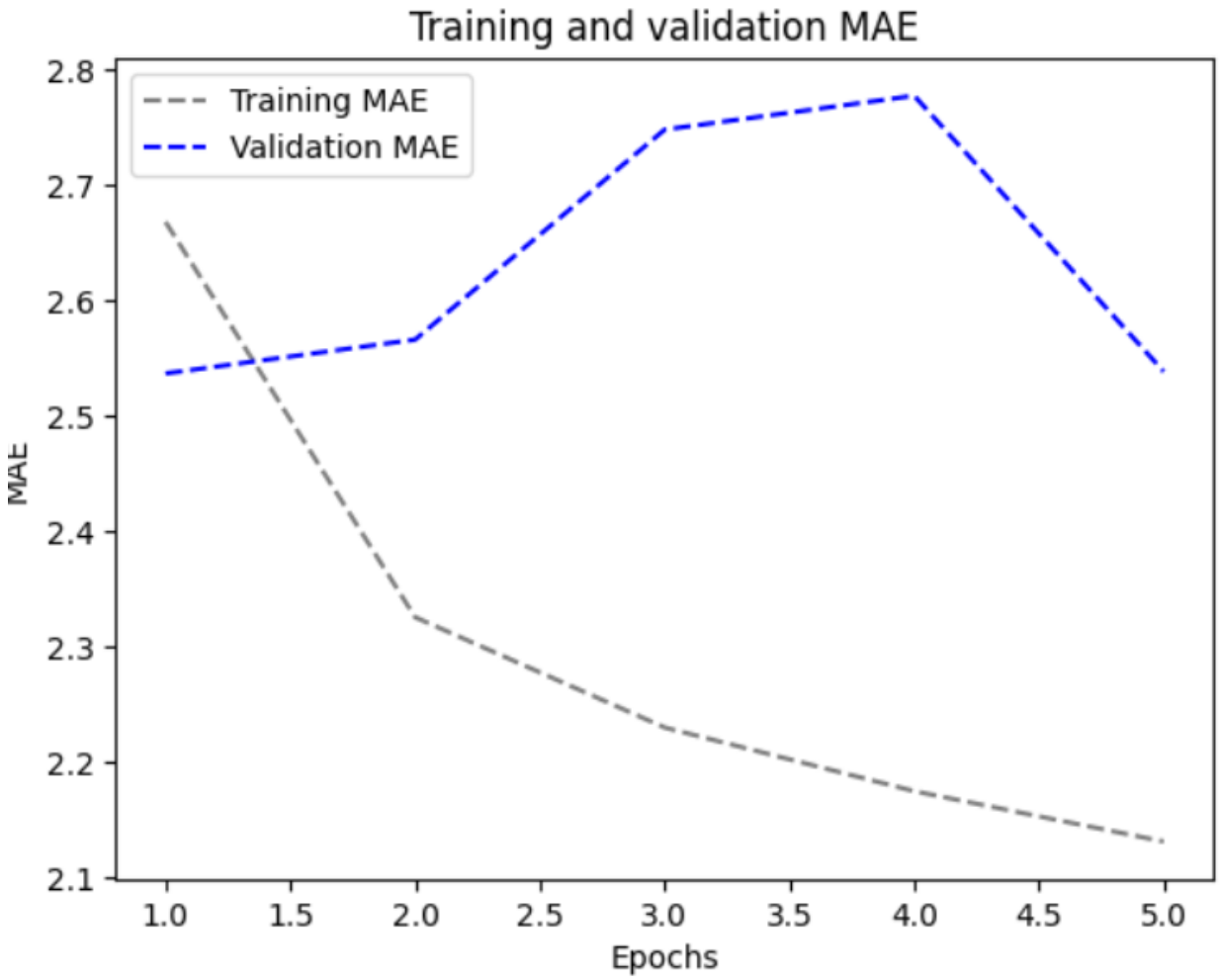
Models used related plots:

Basic Dense Layer: A basic dense layer, also known as a fully connected layer, is a fundamental component of neural networks where each neuron is connected to every neuron in the

previous layer. This type of layer performs a weighted sum of its inputs and applies an activation function to produce its output

This process demonstrates how to build a simple neural network for time series forecasting, emphasizing the importance of validation during training and evaluating the final model on a separate test set.

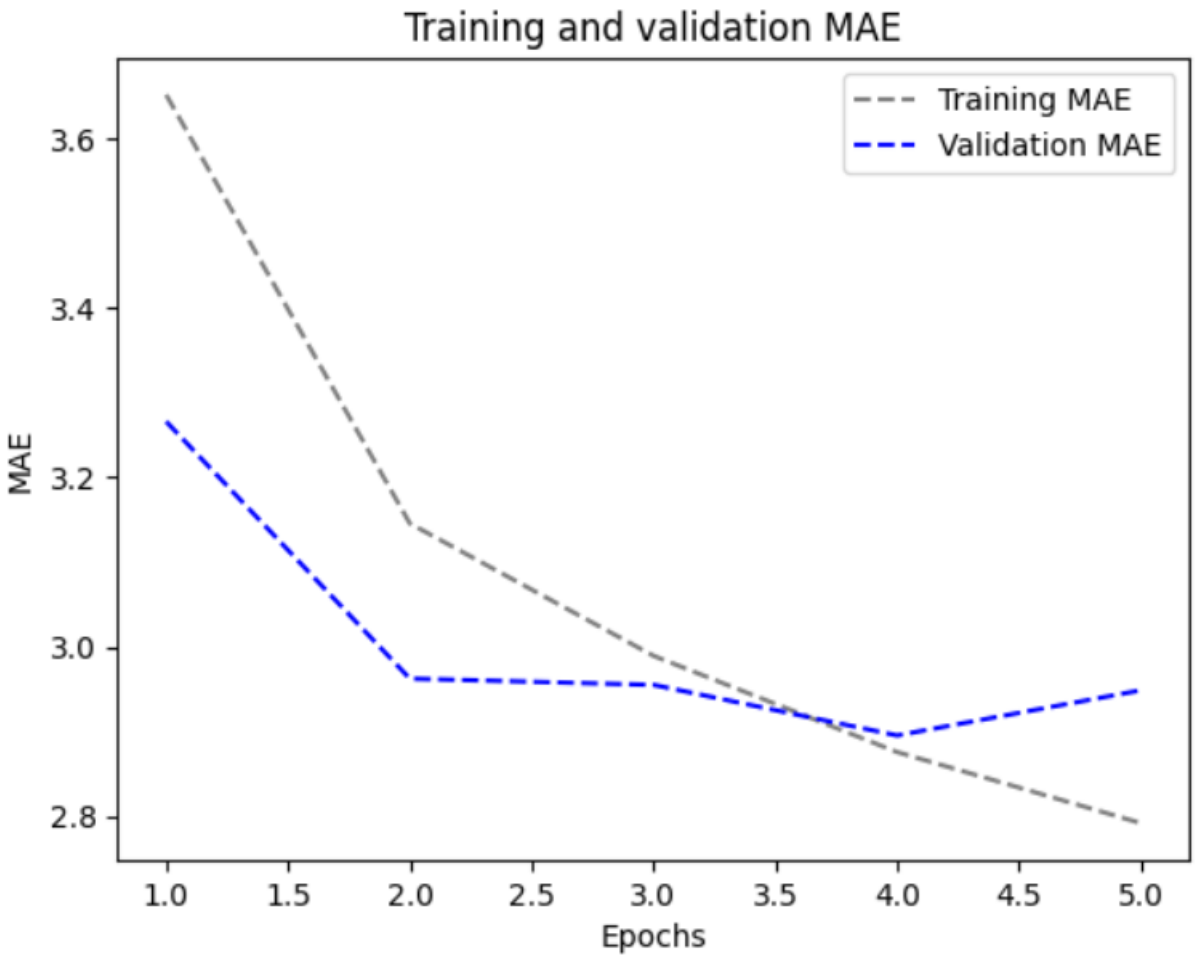
Plot:



1D Convolutional Model:

Using appropriate architectural priors, a convolutional model is suitable for input sequences composed of daily cycles. Like how a temporal convolutional network can use the same representations across several days, a spatial convolutional network reuses representations in

different locations within an image. However, the significantly lower performance of this model compared to the densely connected model is because not all meteorological data meets the translation invariance assumption. Consequently, the convolutional model only achieves a validation MAE of around 2.9 degrees, which is considerably higher than the reasonable baseline.



A Simple RNN:

Recurrent neural networks (RNNs) excel at incorporating information from previous time steps when making decisions, enabling them to detect complex patterns and relationships in sequential data. The internal state of an RNN acts as a memory for past inputs, allowing it to handle sequences of varying lengths. Although a basic RNN theoretically can retain information from all previous time steps, practical challenges arise, such as the vanishing gradient problem,

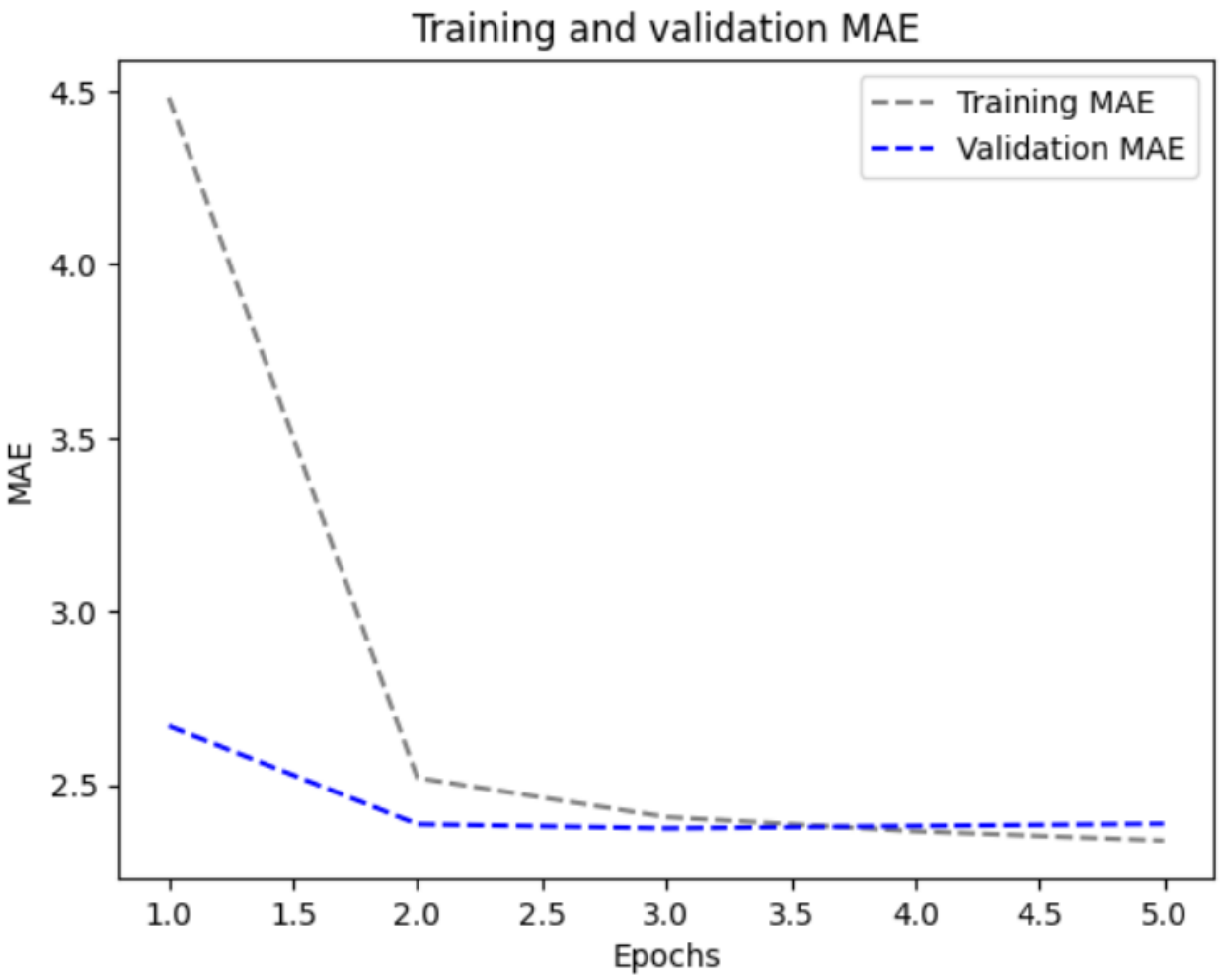
which complicates the training of deep networks. Consequently, as illustrated in the graph, the simplest RNN performs the poorest.

To address this issue, we need to develop LSTM and GRU RNNs using Keras.

Gated Recurrent Unit (GRU):

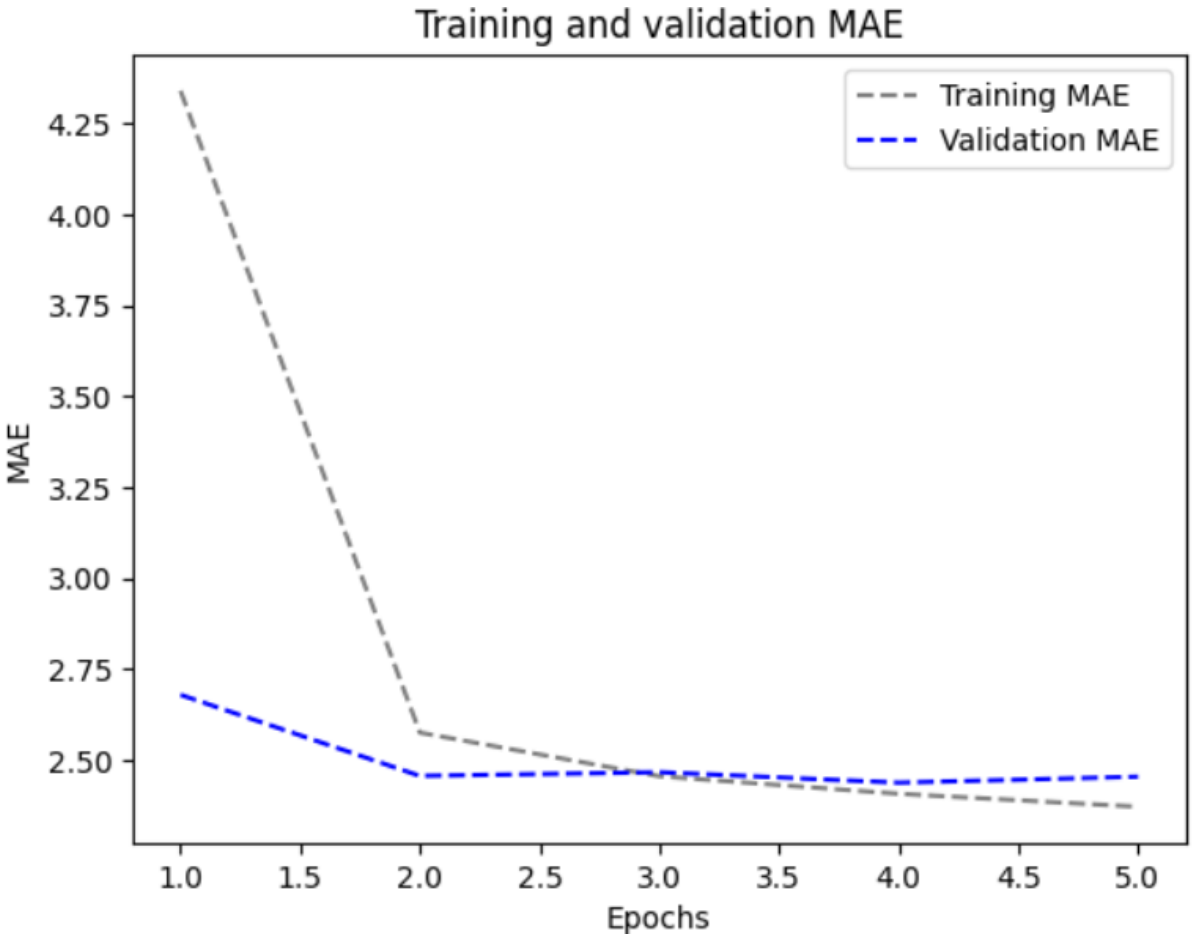
Gated Recurrent Unit (GRU) layers will be utilized instead of LSTM layers. Since GRU and LSTM architectures are quite similar, you can consider GRU as a more streamlined and simplified version of LSTM.

The model with a Test MAE of 2.51 proved to be the most effective. It is less computationally expensive than Long Short-Term Memory (LSTM) models and effectively captures long-range dependencies in sequential data, outperforming the other models.



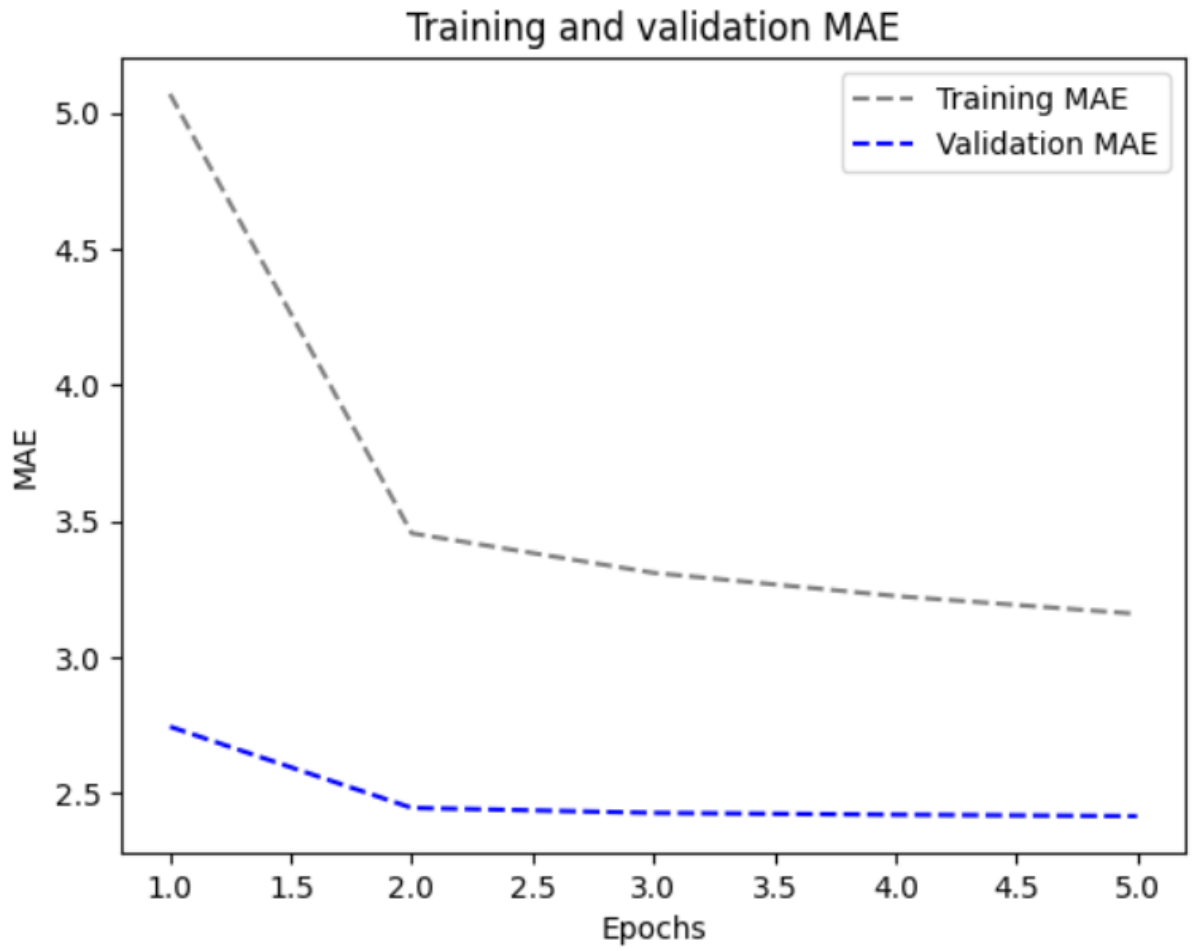
Long Short-Term Memory (LSTM):

Recurrent neural networks provide topologies specifically tailored for this use case, with the Long Short-Term Memory (LSTM) layer being the most widely used. We will soon evaluate the performance of these models, starting with the LSTM layer. The results are notably improved, with a validation MAE as low as 2.47 degrees and a test MAE of 2.57 degrees. Ultimately, the LSTM-based model demonstrates the effectiveness of machine learning in this task by surpassing the common-sense baseline.

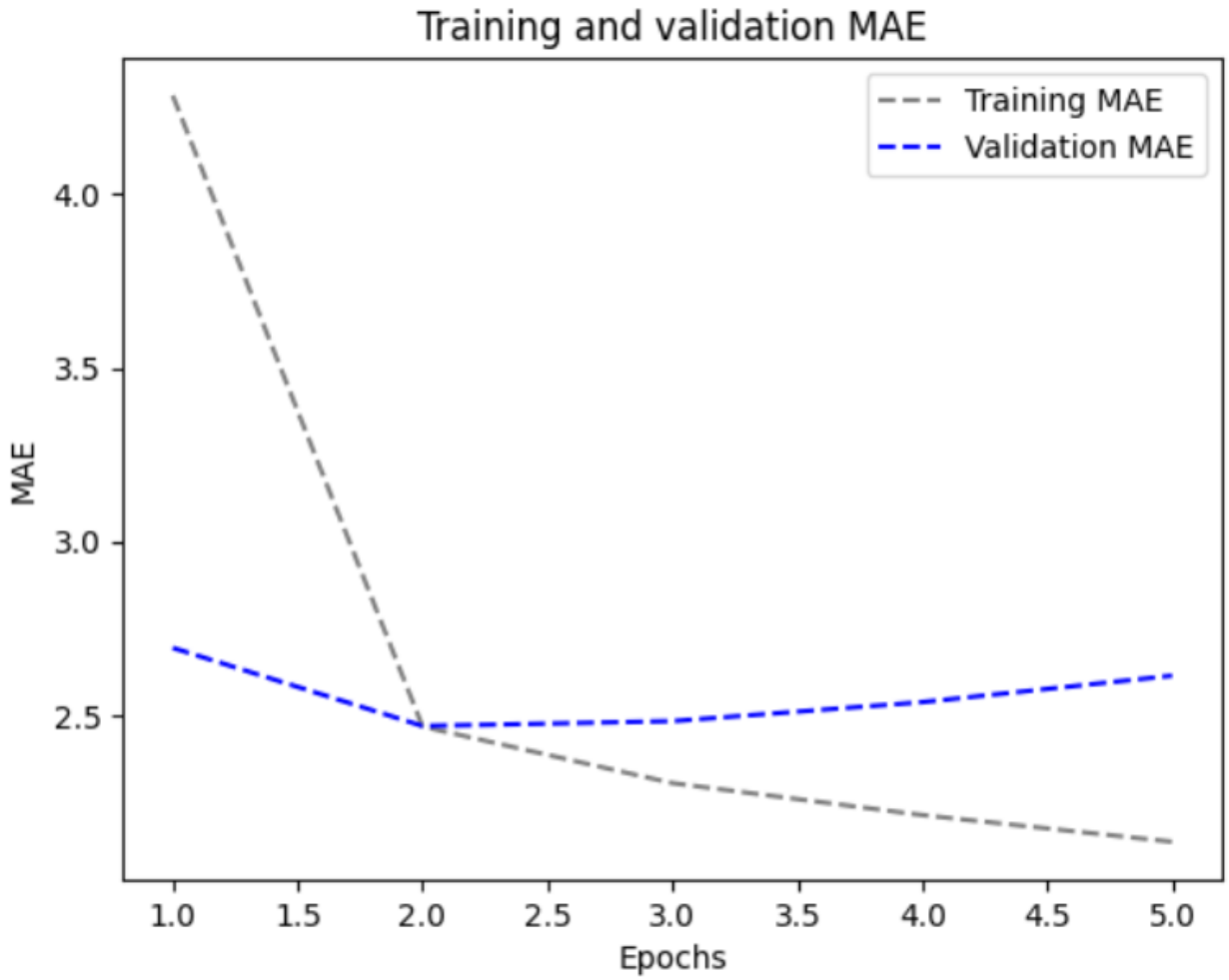


LSTM – Dropout Regularization:

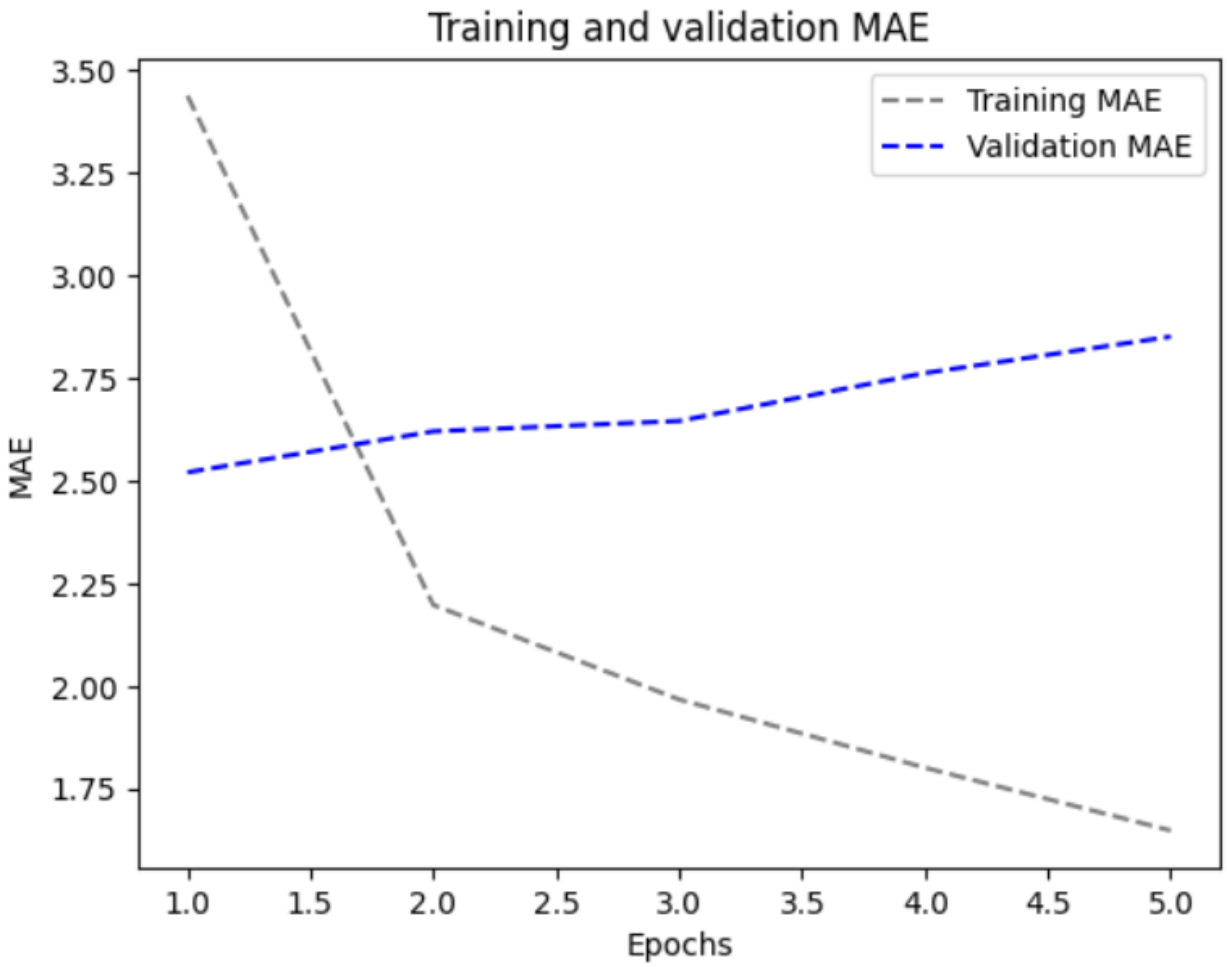
After five epochs, overfitting was no longer an issue. We achieved a test MAE of 2.6 degrees and a validation MAE as low as 2.4 degrees, which is quite satisfactory. I developed six different LSTM models by varying the number of units in the stacked recurrent layers to 8, 16, and 32.



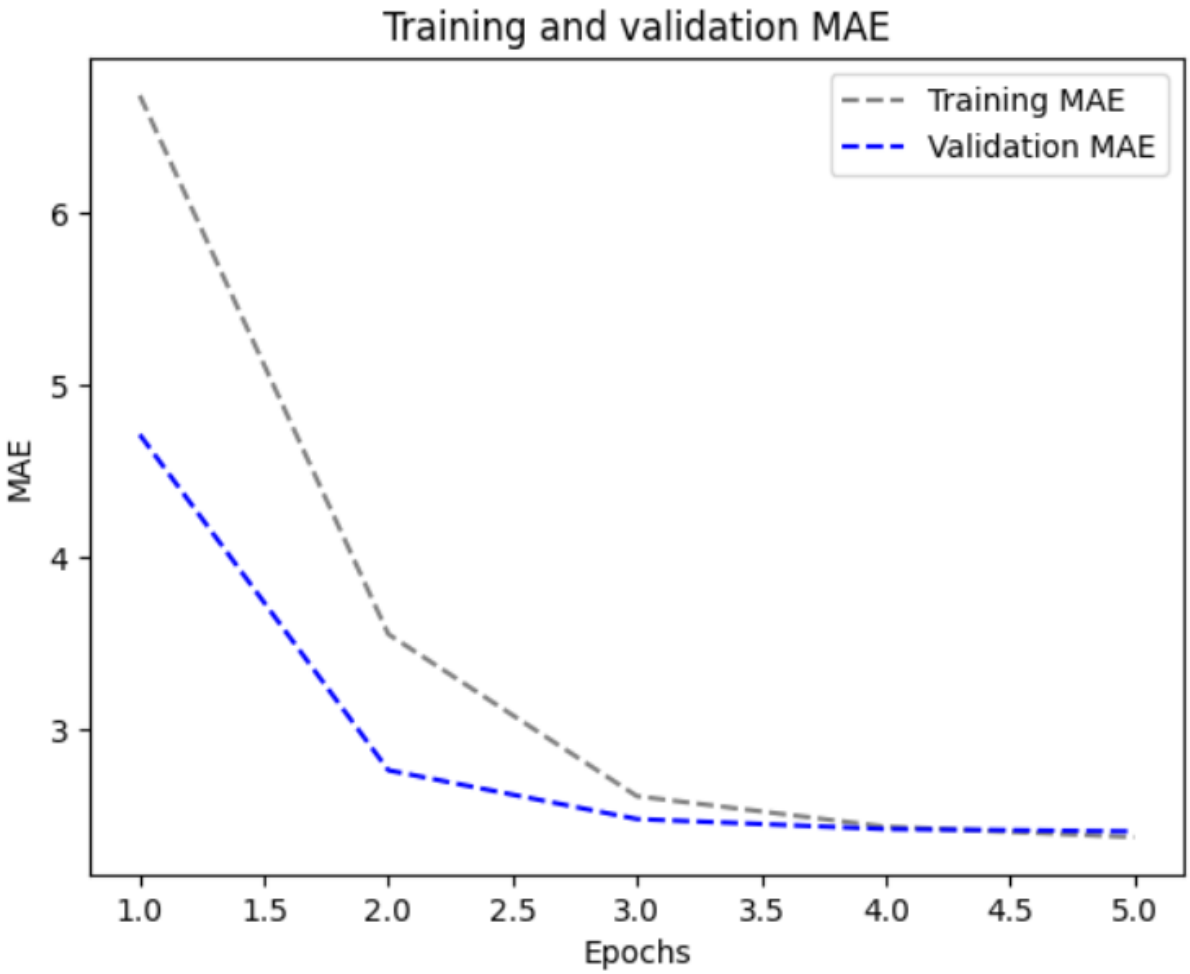
LSTM - Stacked setup with 16 units:



LSTM - Stacked setup with 32 units:

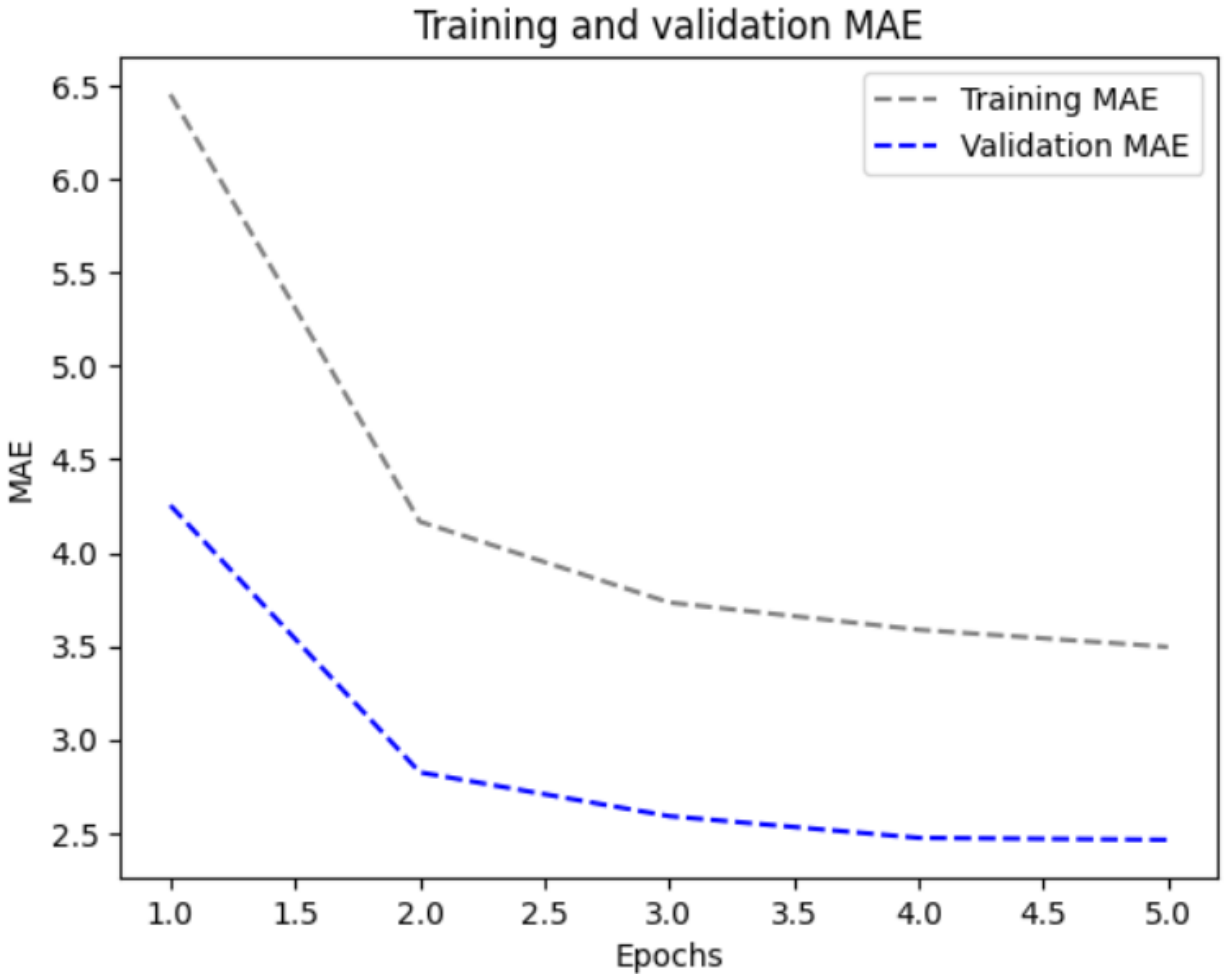


LSTM - Stacked setup with 8 units:

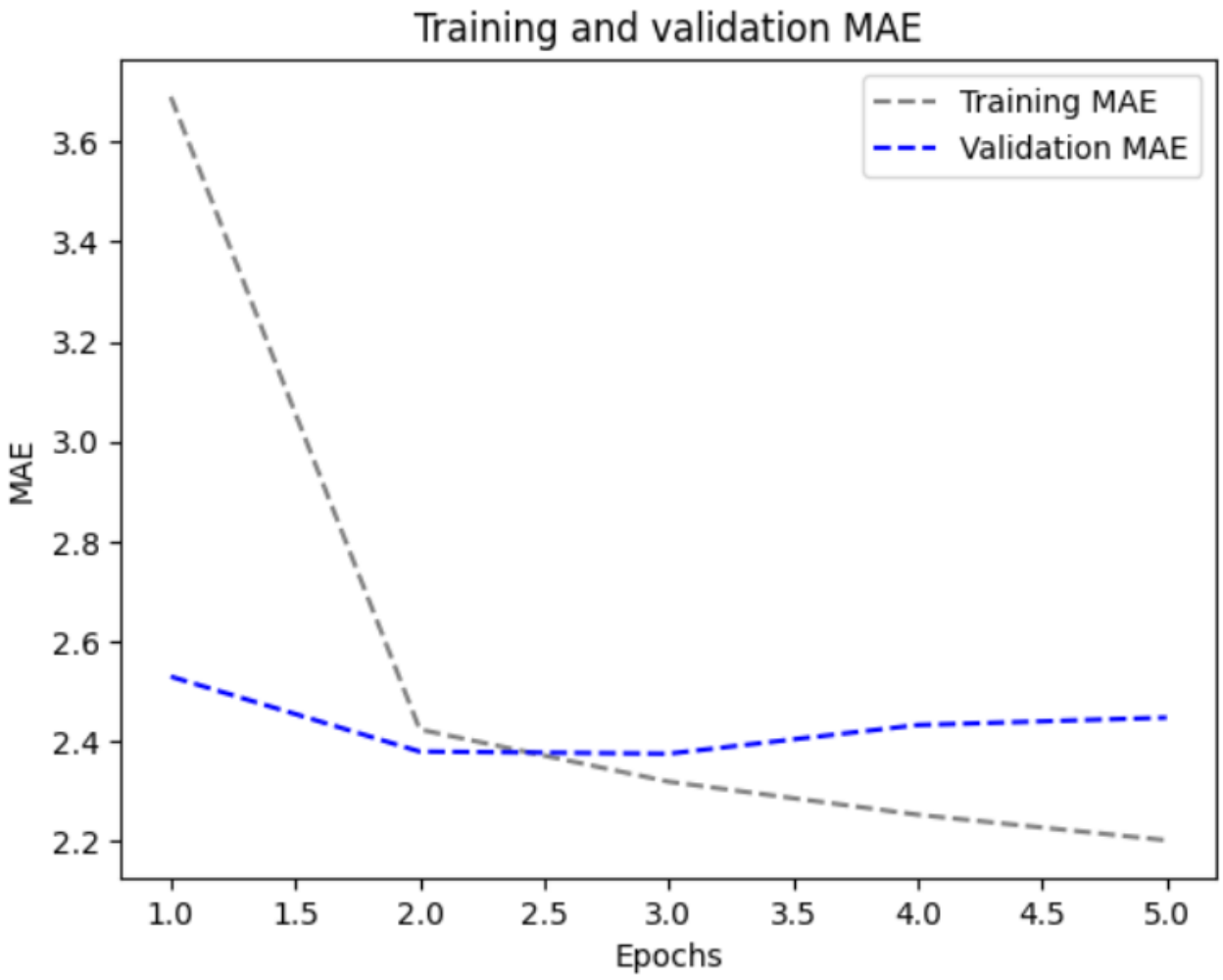


The 8-unit configuration proved to be the most effective, achieving an MAE score of 2.58. To further enhance the model, I experimented with bidirectional data, which presents information to the recurrent network in multiple ways and incorporated recurrent dropout to combat overfitting. These adjustments consistently resulted in lower MAE values compared to the common-sense model, as shown by the MAE assessment graph, and the improvements led to similar MAE values across all models. This is especially notable.

LSTM - dropout-regularized, stacked model:

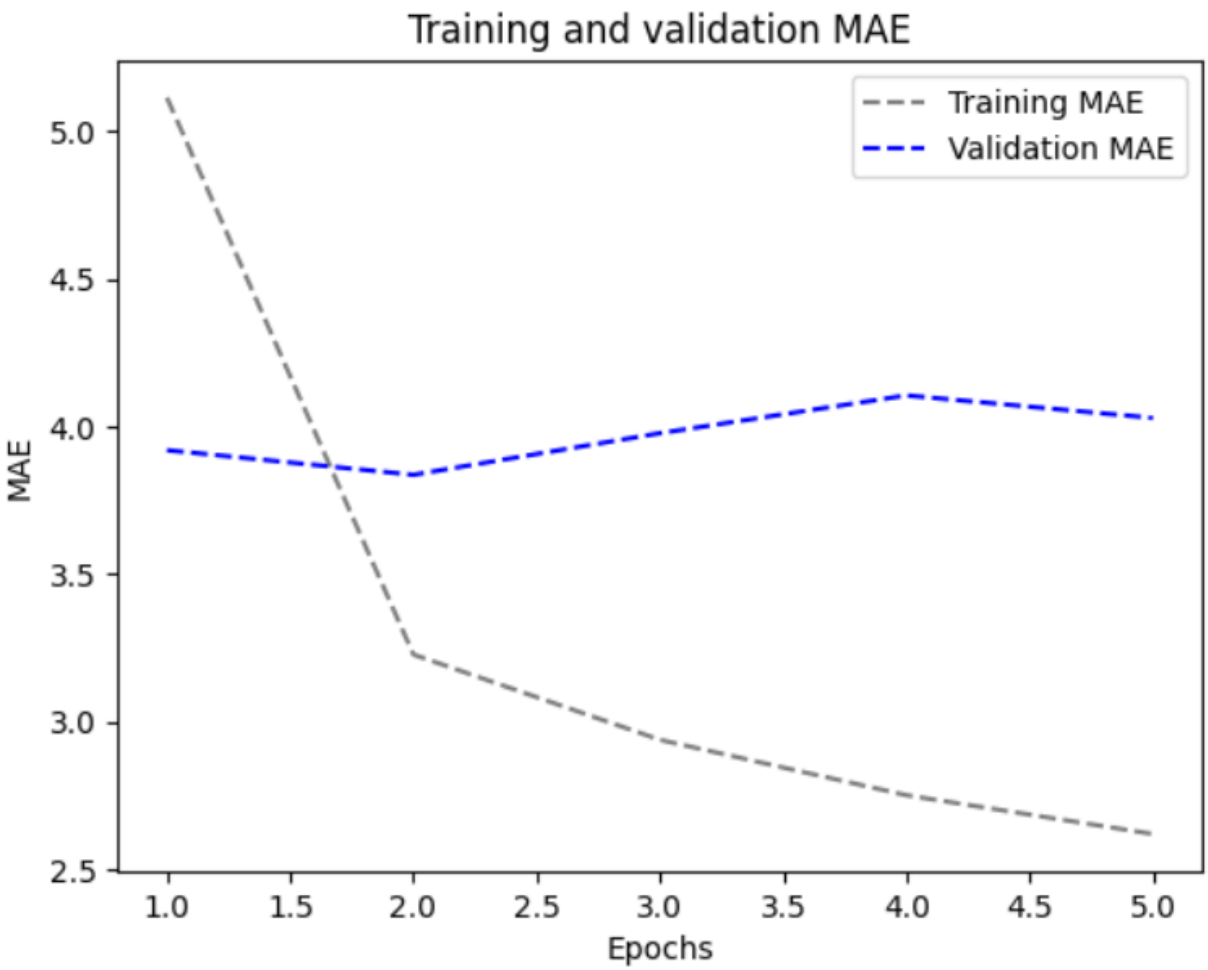


Bidirectional LSTM:

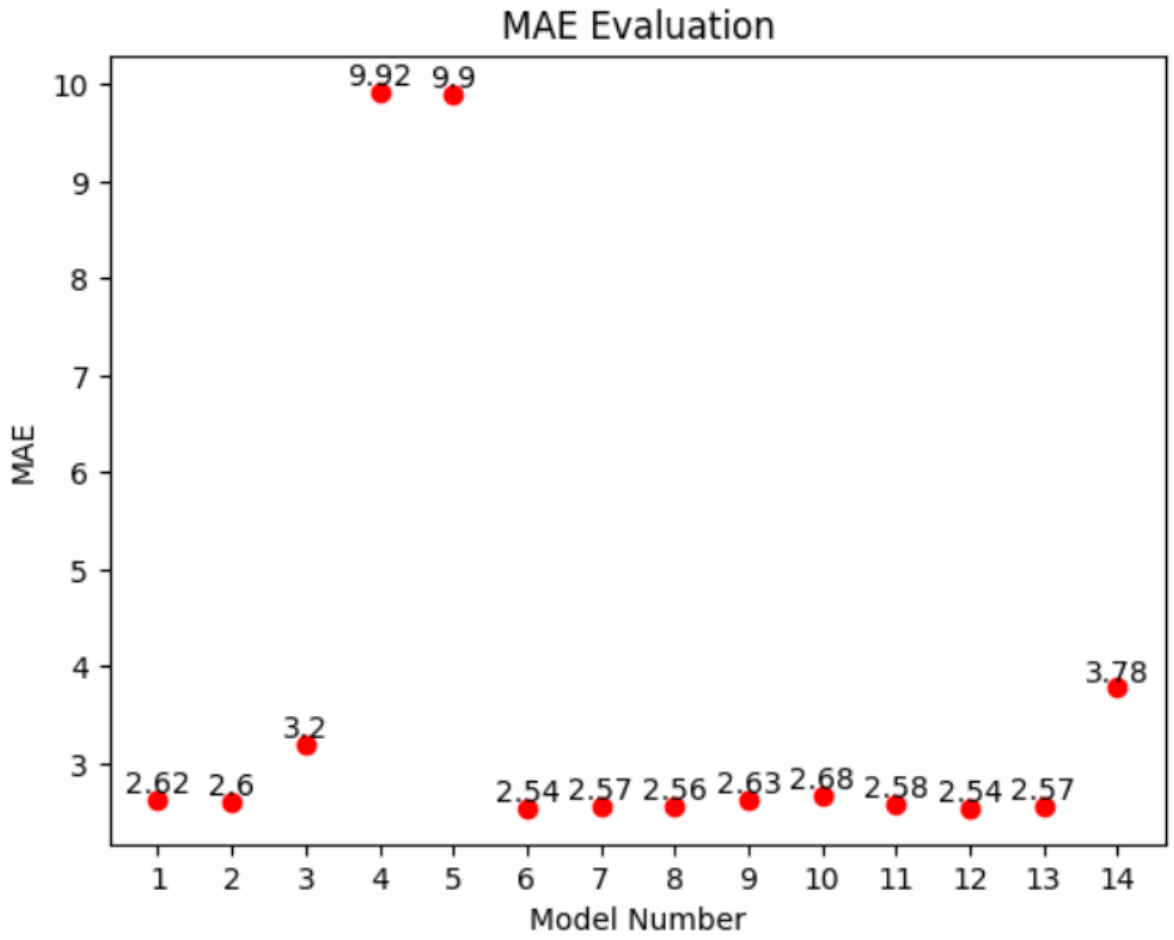


LSTM combined with 1D Convnets:

3.84 MAE was an insufficient result for the model I developed, which combined RNN and 1D convolution. The possible reason for this poor performance is that the convolution limit is destroying the information order



Performance of Every Model:



Model	Validation MAE	Test MAE
Dense Model	2.5385	2.65
1D Convolutional Model	2.9485	3.17
Simple RNN	9.8538	9.92
Stacked Simple RNN	9.8422	9.90
GRU	2.3874	2.51
LSTM Simple	2.4537	2.58
LSTM – dropout Regularization	2.4148	2.63
LSTM – Stacked 16 units	2.6135	2.62
LSTM – Stacked 32 units	2.8507	2.65
LSTM – Stacked 8 units	2.4121	2.54

LSTM – dropout Regularization, stacked model	2.4643	2.63
Bidirectional LSTM	2.4470	2.51
1D Convolutional and LSTM	4.0286	4

Key Findings

- **Best Performing Models**:
  - **GRU**: Achieved the lowest validation MAE of 2.3874 and a test MAE of 2.51.
  - **Bidirectional LSTM**: Also performed well with a validation MAE of 2.4470 and a test MAE of 2.51.
- **Good Performance**:
  - **LSTM Models**: Generally performed well with the best simple LSTM having a validation MAE of 2.4537 and a test MAE of 2.58. The LSTM with 8 stacked units showed similar performance with a validation MAE of 2.4121 and a test MAE of 2.54.
  - **LSTM with Dropout Regularization**: Validation MAE of 2.4148 and a test MAE of 2.63.
  - **Dense Model**: Performed decently with a validation MAE of 2.5385 and a test MAE of 2.65.
- **Poor Performance**:
  - **1D Convolutional Model**: Validation MAE of 2.9485 and a test MAE of 3.17.
  - **1D Convolutional and LSTM Combination**: Performed poorly with a validation MAE of 4.0286 and a test MAE of 4.00.
  - **Simple and Stacked RNNs**: Both configurations had very high MAE scores around 9.9, indicating significant underperformance.

The results indicate that advanced RNN architectures, specifically GRU and Bidirectional LSTM, are the most suitable for time-series weather forecasting tasks. Adjusting hyperparameters and

incorporating regularization techniques can further optimize performance, while convolutional methods and simple RNNs are less effective for this type of data.

In summary, my observations suggest that LSTM and GRU (advanced RNN architectures) are more effective compared to using RNNs combined with 1D convolution, which yielded less satisfactory results. Although Bidirectional LSTM is a popular choice, my tests indicate that GRU is a more efficient approach for managing time series data.

To further optimize GRU performance, it is advisable to adjust hyperparameters such as the number of units in the stacked recurrent layers, the recurrent dropout rate, and the use of bidirectional data.