Time Series Summary

Geethika Vulli
811290653

Using this temperature-forecasting exercise, we will demonstrate the key distinctions between timeseries data and the other dataset types we've already worked with. we will observe that convolutional and highly connected networks are insufficient to handle this kind of dataset, while recurrent neural networks (RNNs), a new form of machine learning approach, absolutely thrive at solving this kind of issue.

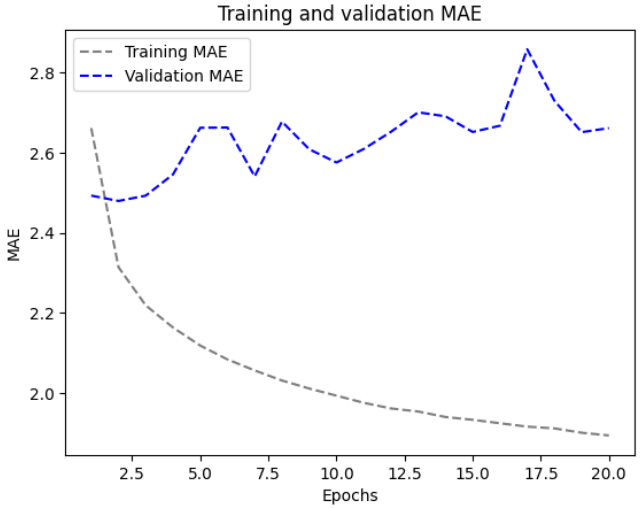
A total of 50% of the data will be used for training, 25% for validation, and the remaining 25% for testing in all our studies. Since our goal is to predict the future based on the past rather than the contrary, it is crucial to use validation and test data that is more recent than the training data when dealing with timeseries data. The validation/test splits should reflect this.
The following will be the precise formulation of the issue: In a day, is it possible to estimate the temperature based on data collected once an hour for the last five days?

Let us start by preprocessing the data so that a neural network can learn to use it. Simple enough—we do not need to perform any vectorization because the data is already numerical.

To analyze time series data, we constructed a total of 14 models. The first model relied on common sense methods and yielded a Mean Absolute Error (MAE) of 2.44 as a baseline. When we created a basic machine learning model with a thick layer, the MAE of 2.62 was somewhat higher. The performance of the thick layer model was poor as the time series data was flattened and the temporal context was lost. While not consistently, some of the validation losses are around the no-learning baseline.

Models :

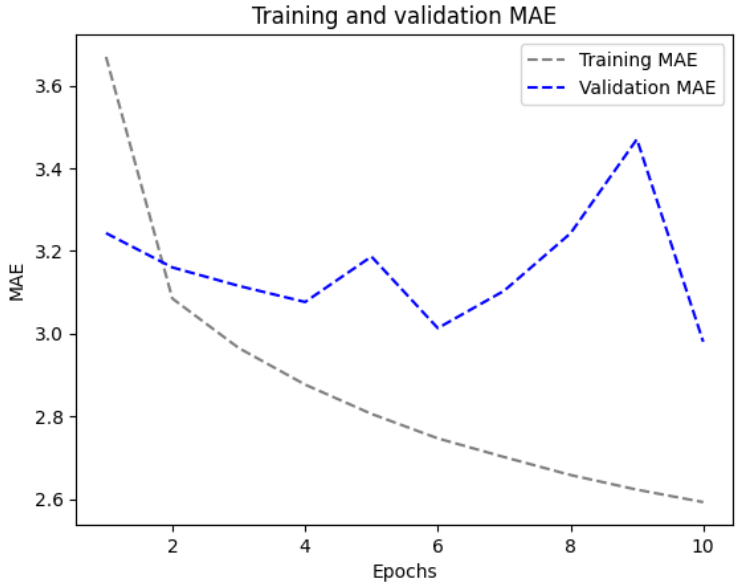
Basic Dense layer



1D convolutional model

In reference to utilizing appropriate architectural priors, given that the input sequences consist of daily cycles, a convolutional model would be suitable. As a spatial convolutional network may reuse the same representations across multiple places in an image, so too might a temporal convolutional network use the same representations over different days.

Indeed, this model's performance is considerably poorer than that of the densely linked model; it only manages a validation MAE of around 2.9 degrees, which is far from the sensible baseline, because not all meteorological data adheres to the translation invariance assumption.



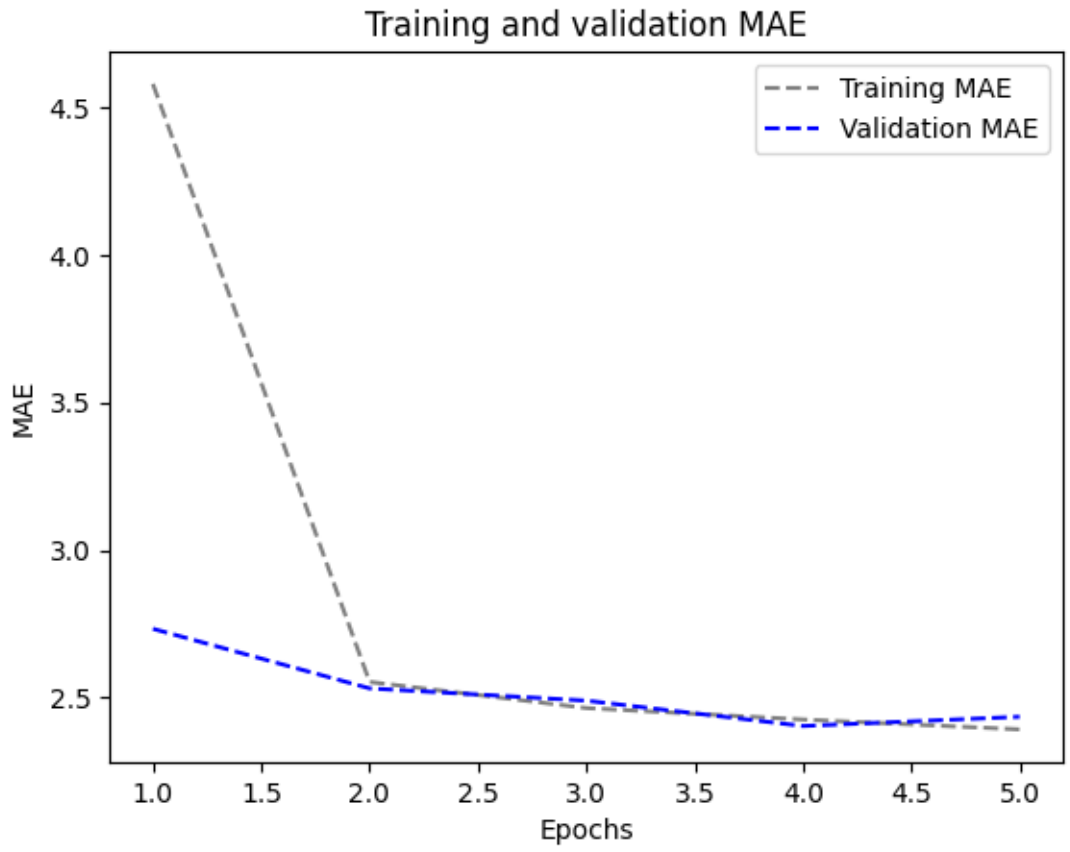
A Simple RNN

Recurrent neural networks (RNNs) has the exceptional capacity to incorporate historical time step information into present-day decision-making procedures, enabling them to discern complex associations and trends in sequential data. It is possible to describe sequences of different lengths since an RNN's internal state functions as a memory of previous inputs. Practical difficulties arise even though a simple RNN may theoretically preserve data from all prior times. This causes training for deep networks to be challenging due to the vanishing gradient problem. Furthermore, the graph indicates that the simplest RNN performs the poorest out of all of them.

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

GRU

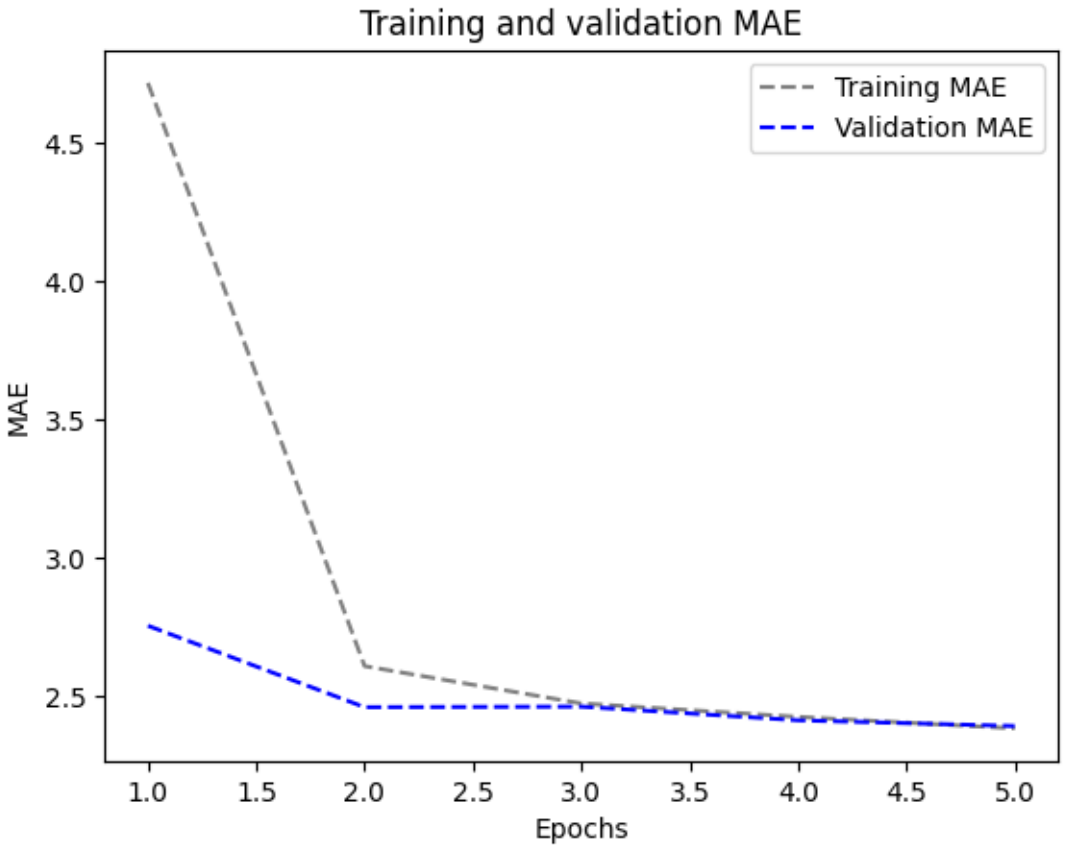
Instead of LSTM layers, we'll utilize Gated Recurrent Unit (GRU) layers. GRU and LSTM are quite similar; consider it an abridged, more straightforward form of the LSTM architecture.



We achieved Test MAE – 2.54 is discovered to be the most effective model, is less computationally costly than Long Short-Term Memory (LSTM) models and effectively captures long-range dependencies in sequential data when compared to the other models.

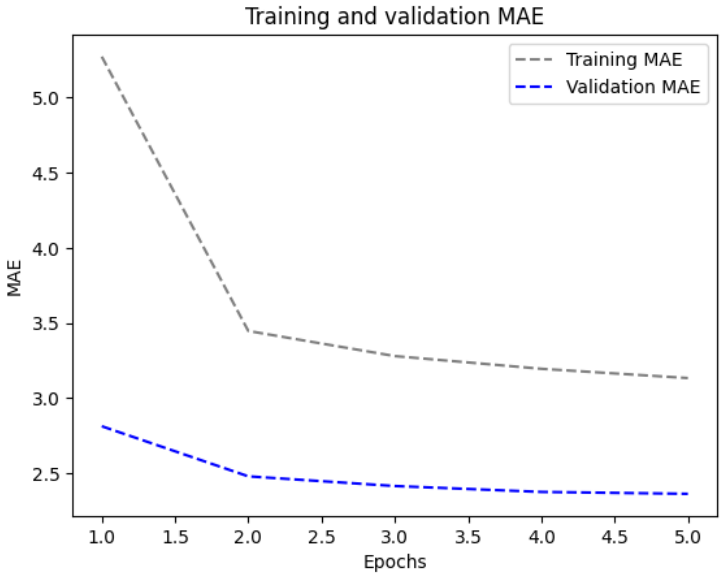
LSTM

Recurrent neural networks provide a class of neural network topologies particularly created for this use application. One of the most well-liked of them is the Long Short-Term Memory (LSTM) layer. Let us test out the LSTM layer first, and we will see how these models function in a moment.



Much improved! We obtain a test MAE of 2.57 degrees and a validation MAE as low as 2.39 degrees. Finally, the LSTM-based model outperforms the common-sense baseline (albeit only somewhat, at least for the time being), highlighting the effectiveness of machine learning in this endeavor.

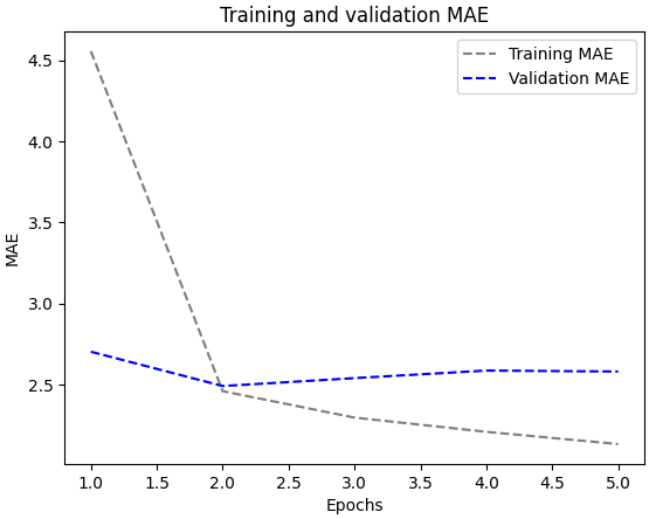
LSTM - dropout Regularization



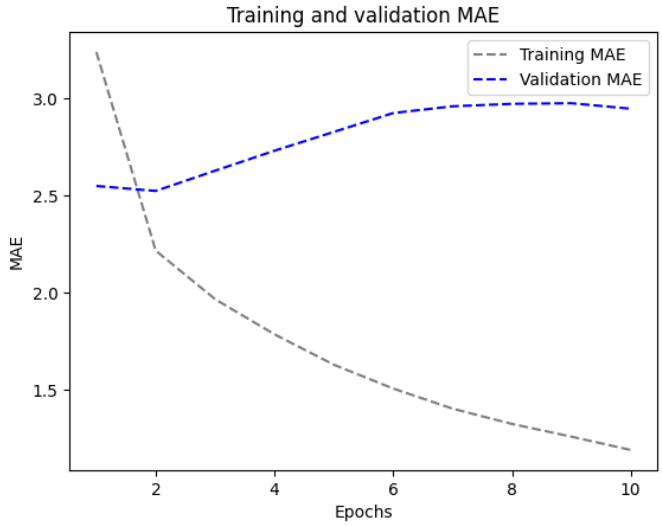
Achievement! The overfitting that occurred over the first 5 epochs has stopped. We attain a test MAE of 2.56 degrees and a validation MAE as low as 2.36 degrees. Not too horrible.

Using 8, 16, and 32 units as varying numbers of units inside the stacked recurrent layers, I constructed six distinct LSTM models.

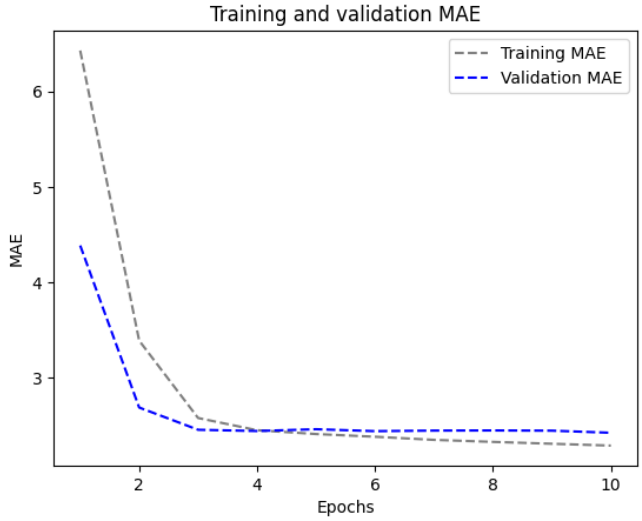
LSTM - Stacked setup with 16 units



LSTM - Stacked setup with 32 units



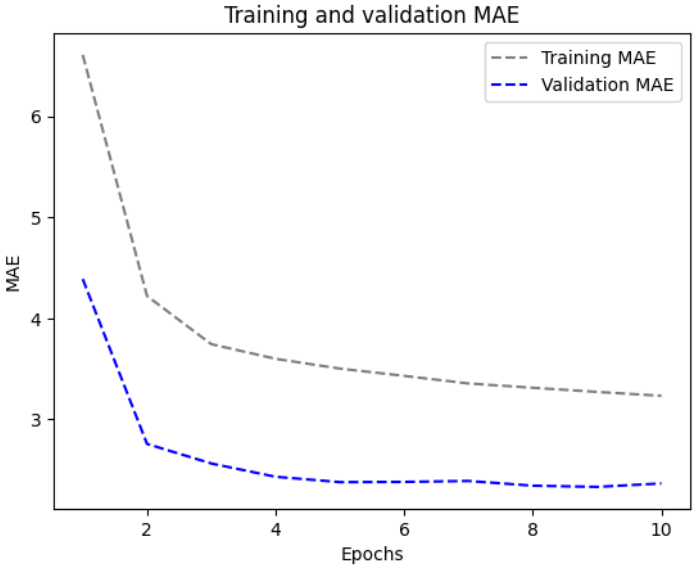
LSTM - Stacked setup with 8 units



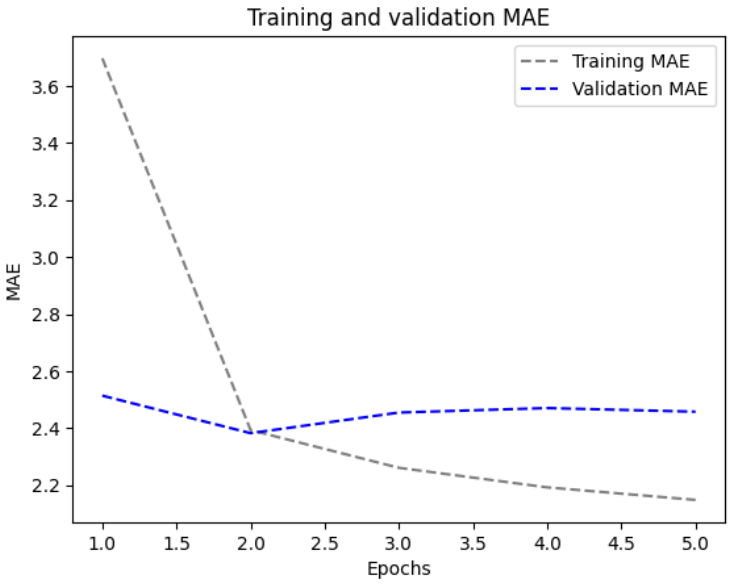
With an MAE score of 2.58, the 8-unit configuration among these alternatives showed the best performance.

To enhance the model, I also experimented with methods like recurrent dropout to counteract overfitting and used bidirectional data, which lowers MAE values by presenting information to a recurrent network in a variety of ways. These improvements produced comparable MAE values across the board for all models, and most remarkably, these values were consistently lower than those of the common-sense model, something that the MAE assessment graph also confirms.

LSTM - dropout-regularized, stacked model

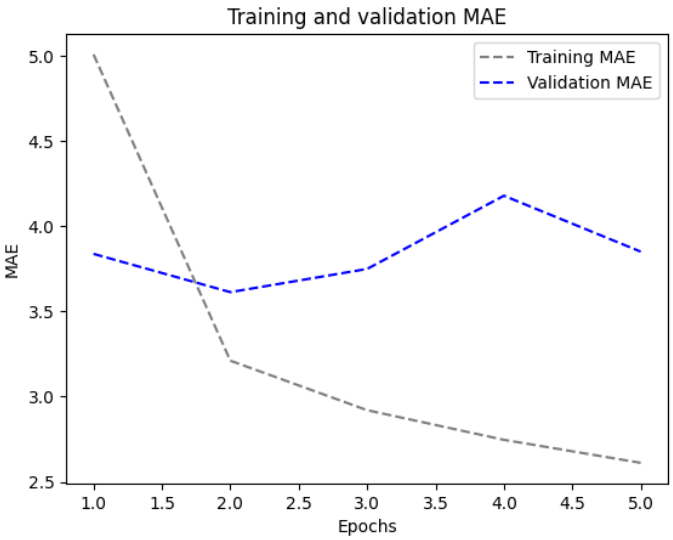


Bidirectional LSTM

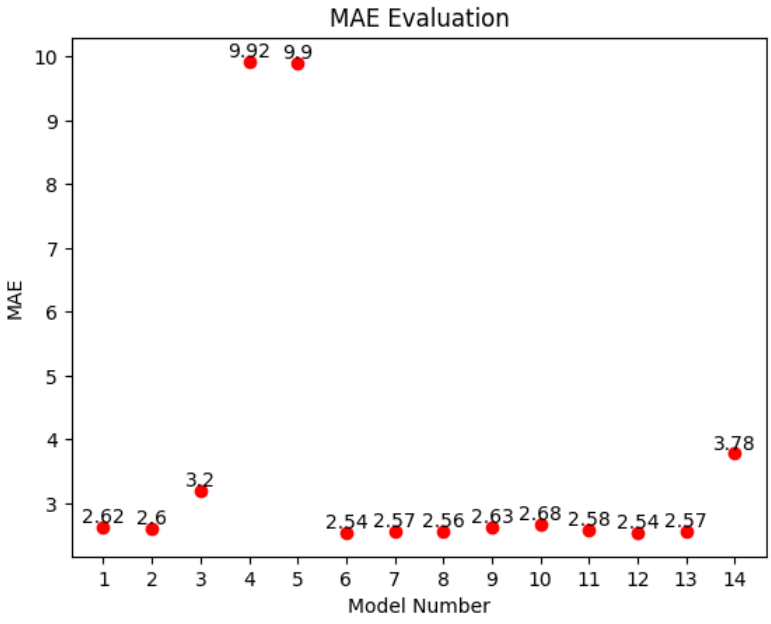


1D Convnets and LSTM together

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 summary, my observations suggest that utilizing LSTM and GRU (advanced RNN architectures) is the better option, whereas combining RNN with 1D convolution produced inadequate results. After some testing, I believe that GRU is a more effective option for handling time series data, even though Bidirectional LSTM is still a popular choice. Hyperparameters that should be adjusted to maximize GRU include the number of units in the stacked recurrent layers, the recurrent dropout rate, and the use of bidirectional data.