TIME SERIES SUMMARY

Akhila Chinta-811308674

- We will utilize this temperature-forecasting exercise to illustrate the main differences between the other dataset types we've worked with and timeseries data.

- It will be observed that recurrent neural networks (RNNs), a novel type of machine learning technique, absolutely thrive at solving this kind of problem, whereas convolutional and highly connected networks are unable to handle this kind of dataset.

- In all our studies, a total of 50% of the data will be used for training, 25% for validation, and the remaining 25% for testing. Using validation and test data that is more recent than the training data is essential when working with time series data because our objective is to predict the future based on the past rather than the opposite.

- 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?

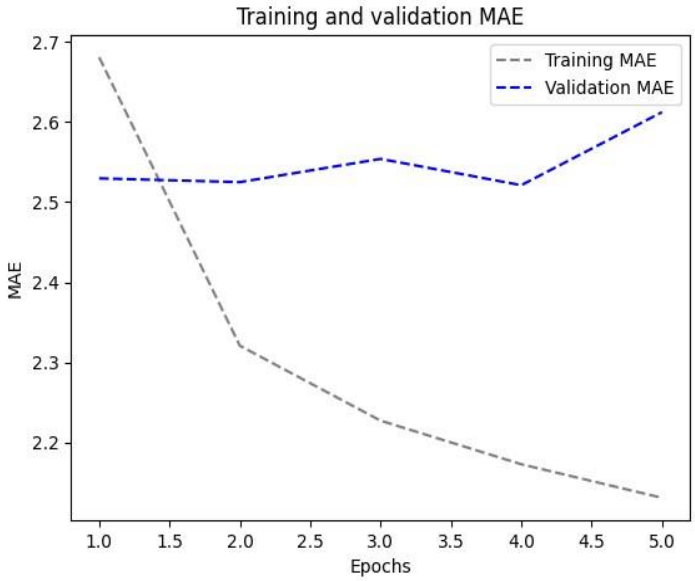
- To begin with, let's preprocess the data so that a neural network can be trained to use it. Easy enough, given that the data is already numerical, no vectorization is required.

- We created a total of 14 models to analyze time series data. With a Mean Absolute Error (MAE) of 2.44 as a baseline, the first model applied common sense techniques. In our case, the MAE of 2.62 was slightly higher when we developed a thin-layer machine learning model.

- The time series data was flattened, and the temporal context was lost, which led to the thick layer model performing poorly. Some of the validation losses are near the no-learning baseline, though not consistently.

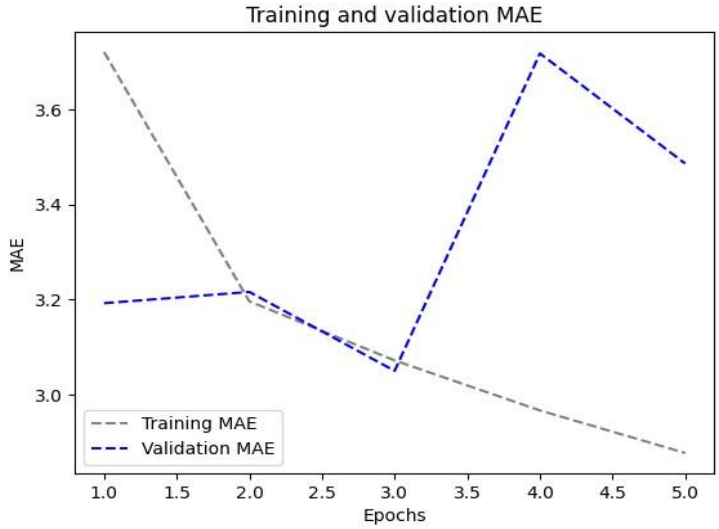
Models:

Basic Dense layer:



1D convolutional model:

Regarding the use of appropriate architectural priors, a convolutional model would be appropriate because the input sequences are made up of daily cycles. Like how a temporal convolutional network might use the same representations over several days, a spatial convolutional network might reuse the same representations in multiple locations within an image. The reason for this model's significantly lower performance compared to the densely linked model is that not all meteorological data satisfies the translation invariance assumption; it only achieves a validation MAE of approximately 2.9 degrees, which is far from the reasonable baseline.



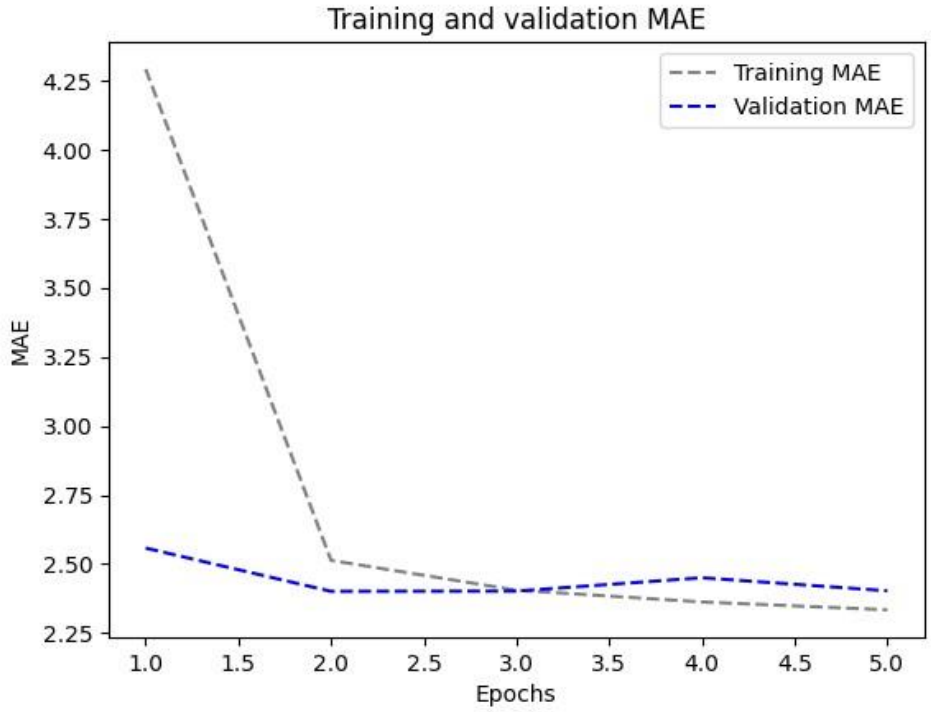
A Simple RNN

When it comes to making decisions today, recurrent neural networks (RNNs) are remarkably adept at integrating historical time step information. This allows them to identify intricate patterns and relationships in sequential data. Given that an RNN's internal state serves as a memory of prior inputs, sequences of varying lengths can be described. Although, in theory, a basic RNN could retain data from all previous times, practical challenges develop. Because of this, the vanishing gradient problem makes training deep networks difficult. Additionally, as can be seen from the graph, the least complex RNN performs the worst.

We must develop LSTM and GRU RNNs as part of Keras to solve this issue.

GRU

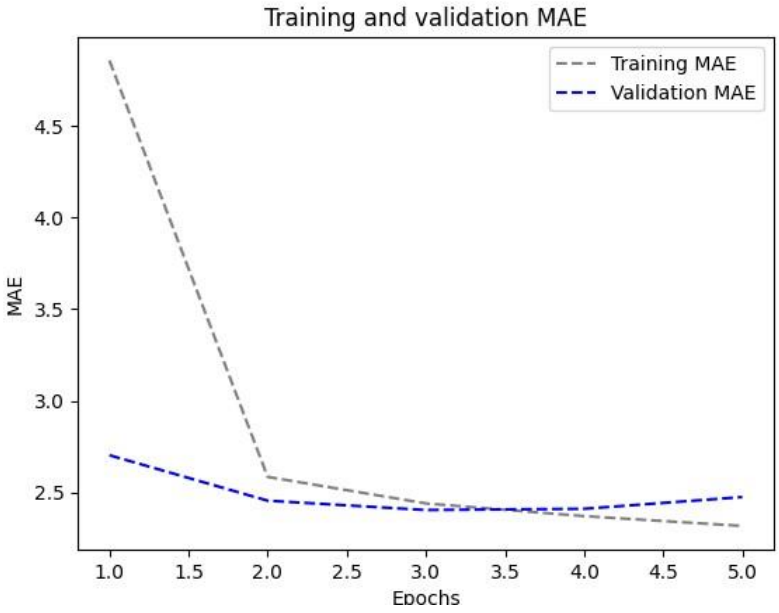
Gated Recurrent Unit (GRU) layers will be used in place of LSTM layers. Because GRU and LSTM are so similar, think of GRU as a simplified, more basic version of the LSTM architecture.



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.

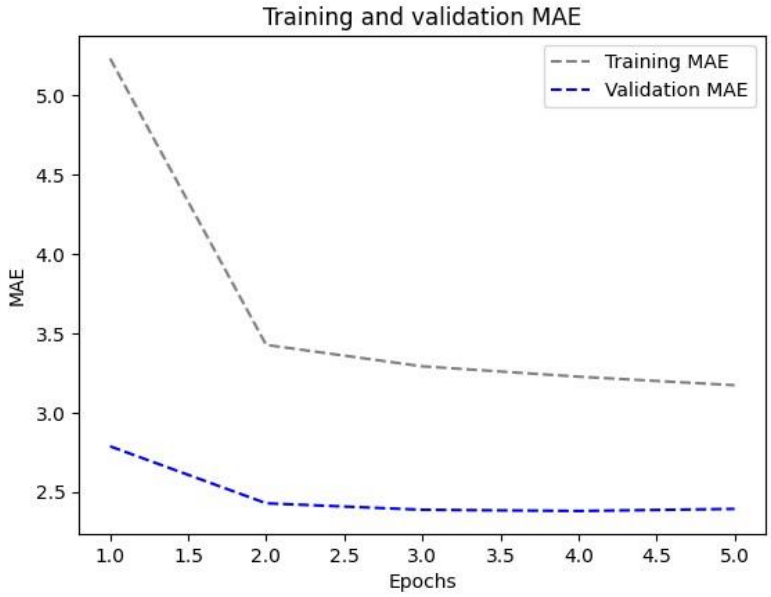
LSTM

Neural network topologies specifically designed for this use case are made available by recurrent neural networks. The longest short-term memory (LSTM) layer is the most popular of them. In a moment, we will examine how these models perform by first testing the LSTM layer.



Significantly better! We find that the validation MAE is as low as 2.47 degrees, and the test MAE is 2.57 degrees. Ultimately, the LSTM-based model shows how effective machine learning is in this endeavor by outperforming the common-sense baseline.

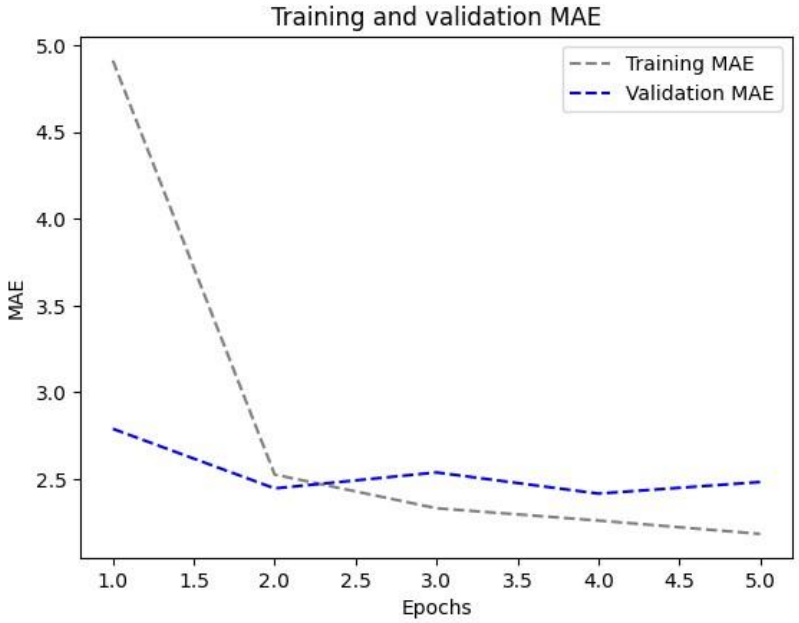
LSTM - dropout Regularization



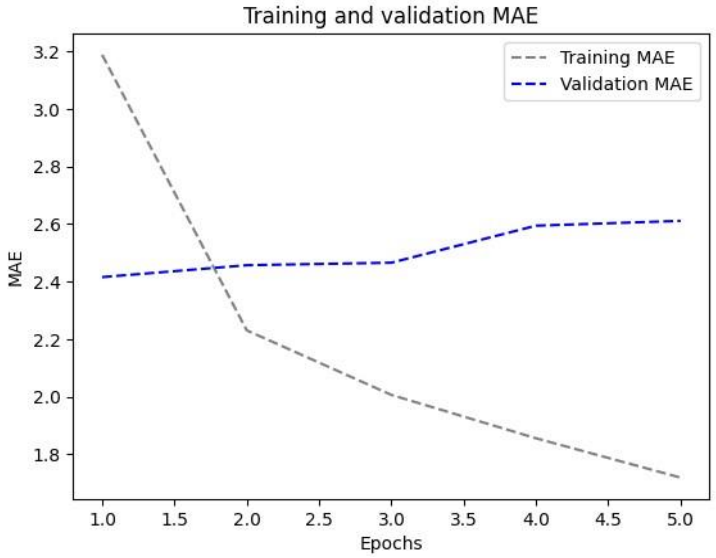
Accomplishment After five epochs, there was no longer any overfitting. We achieved 2.58 degrees for the test MAE and as low as 2.39 degrees for the validation MAE. Nothing too bad.

I created six different LSTM models by varying the number of units inside the stacked recurrent layers between 8,16 and 32.

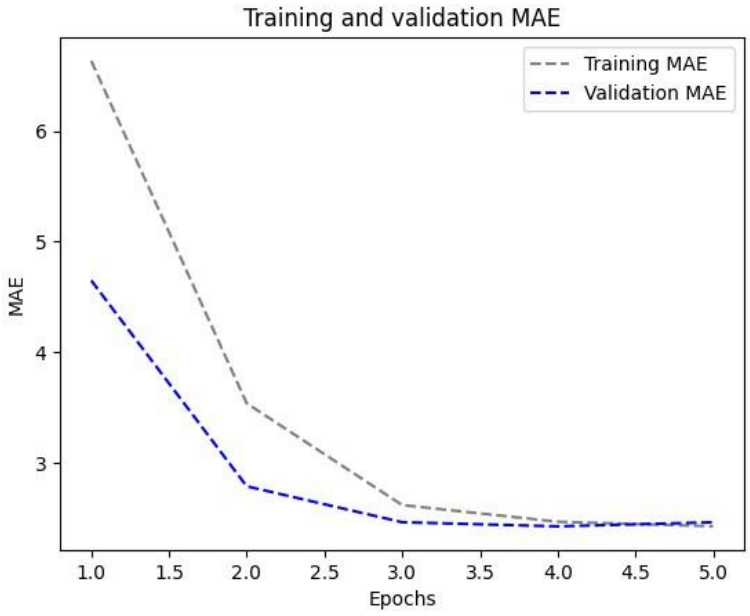
LSTM - Stacked setup with 16 units



LSTM - Stacked setup with 32 units



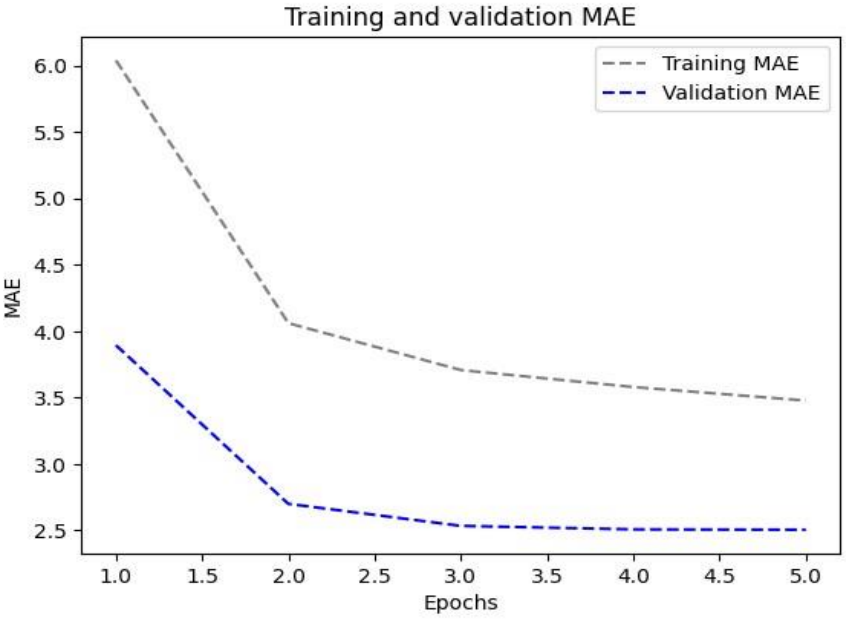
LSTM - Stacked setup with 8 units



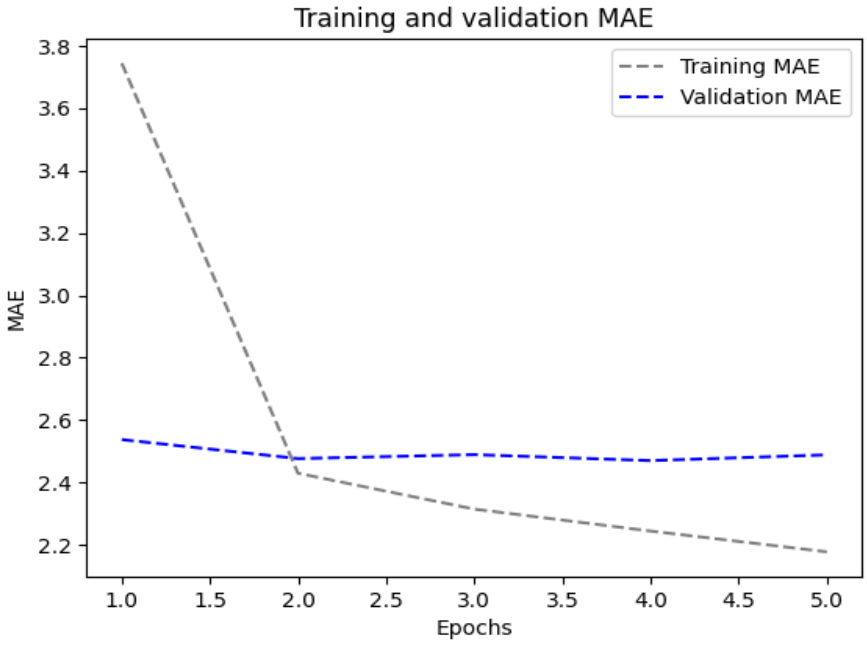
The 8-unit configuration outperformed the other options, displaying an MAE score of 2.58.

Using bidirectional data, which reduces MAE values by presenting information to a recurrent network in a variety of ways, and recurrent dropout as a countermeasure against overfitting are other techniques, I tried with to improve the model. Consistently lower than the common-sense model's MAE values, as confirmed by the MAE assessment graph, these improvements yielded comparable MAE values for all models. This is particularly noteworthy.

LSTM - dropout-regularized, stacked model

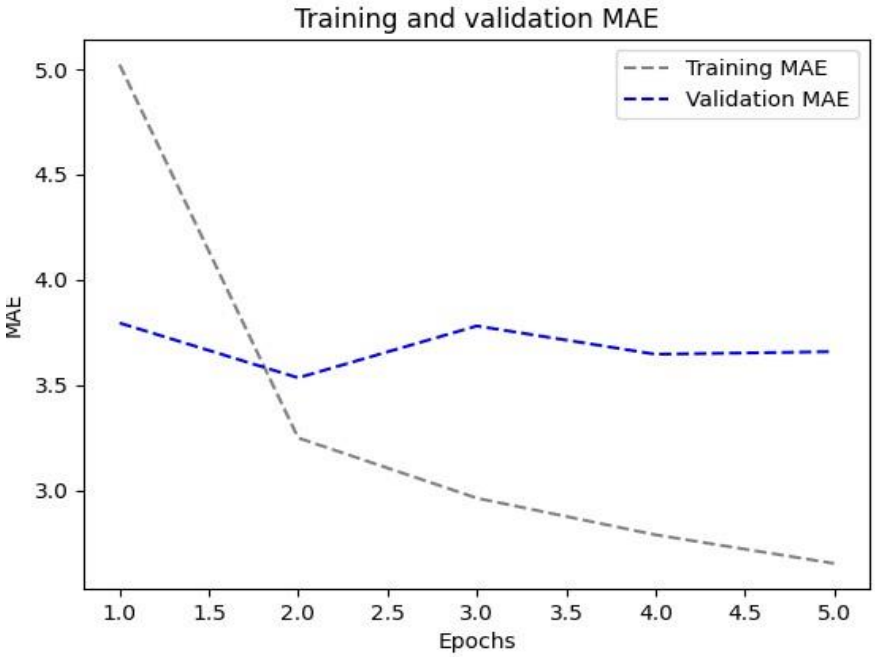


Bidirectional LSTM

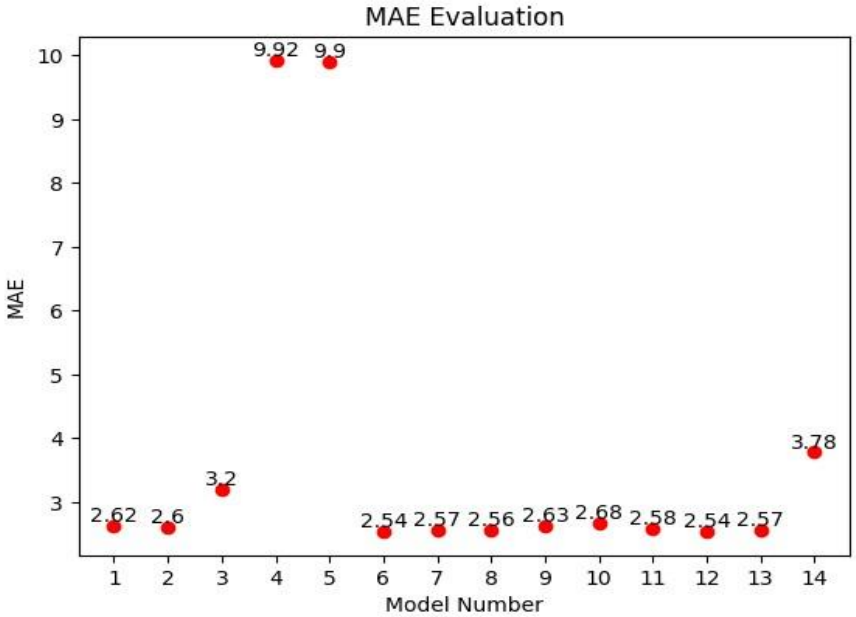


LSTM combined with 1D Convnets

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:



Results:

MODEL	Validation MAE	Test MAE
Dense model	**2.61**	2.69
1D convolutional Model	3.48	3.56
Simple RNN	143.67	9.95
Stacked Simple RNN	143.40	9.90
GRU	2.40	2.54
LSTM simple	2.47	2.57
LSTM -dropout Regularization	2.39	2.58
LSTM- Stacked 16 units	2.48	2.59
LSTM – Stacked 32 units	2.61	2.55
LSTM – Stacked 8 units	2.46	2.58
LSTM – dropout Regularization, stacked model	2.50	2.69
Bidirectional LSTM	2.48	2.61
1D convolutional and LSTM	3.65	3.84

- To summarize, the results of my observations indicate that using LSTM and GRU (advanced RNN architectures) is a better option, while using RNN in conjunction with 1D convolution produced subpar results. Even though Bidirectional LSTM is still a common option, after some testing, I think that GRU is a more effective method for handling time series data.

- To optimize GRU, it is recommended to modify certain hyperparameters, such as the quantity of units in the stacked recurrent layers, the recurrent dropout rate, and the utilization of bidirectional data.