Time Series Summary

Tarun Kumar Korimi
811299922

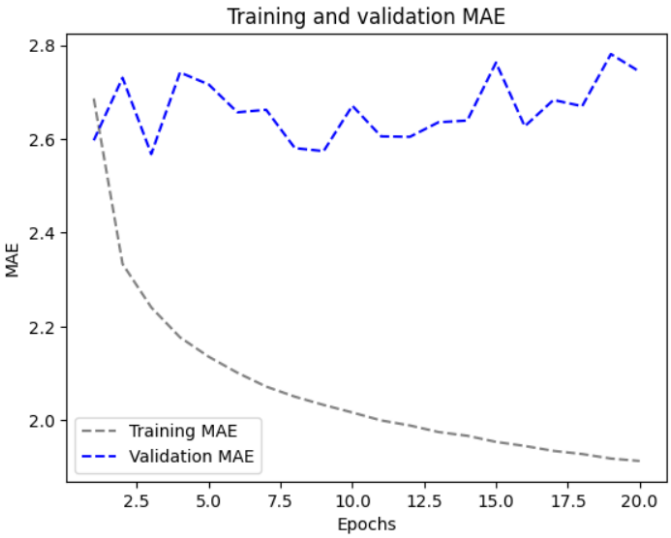
By the end of this temperature-predicting exercise, we will be able to show the important differences of time-series data from the other types of datasets we have been studying up until now. We shall see that convolutional and very connected networks are badly suited for this type of dataset while RNNs, a new kind of machine learning method, are radically better at solving this type of problems.

We will be dividing our data in all these studies into 50% for training, 25% for validation, and the remaining 25% for testing. Since we want to predict the future from the past and not the other way round, it is important that in timeseries data, the validation and test data should be relatively more recent than the training data. This fact should be reflected in the validation/test splits.
The problem would be exactly articulated as: Given one recording taken every hour for five previous days, can one predict the temperature at any instant during a day?

Let's start off by preprocessing the data so a neural network can learn from it. Simple enough—no need to vectorize anything because the data is already numerical:.
We have done a total of 14 models to analyze time series. The first one was using the most basic common-sense methods. It provided us with an MAE of 2.44 to base our future analysis on. Then, after creating a basic machine learning model with a thick layer, the MAE was a bit higher, 2.62. The performance of the thick layer model was very poor: time series data were flattened and temporal context has vanished, whereas some of the validation losses are around no-learning baseline.

Models :

Basic Dense layer

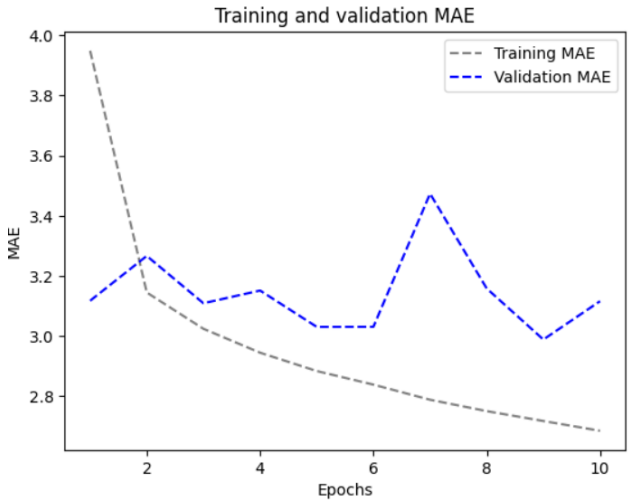


1D convolutional model

Notably, for leveraging appropriate architectural priors, since the input sequences show daily cycles, a convolutional model would be applicable. Just as a spatial convolutional network might reuse the same representation across different locations in an image, so too might a temporal convolutional network make use of the same representations across different days.

Indeed, the performance of this model stands far worse compared to the densely connected model; it turns out to achieve a validation MAE of only about 2.9 degrees, far from the sensible baseline since not all meteorological data

follows translation invariance assumptions.



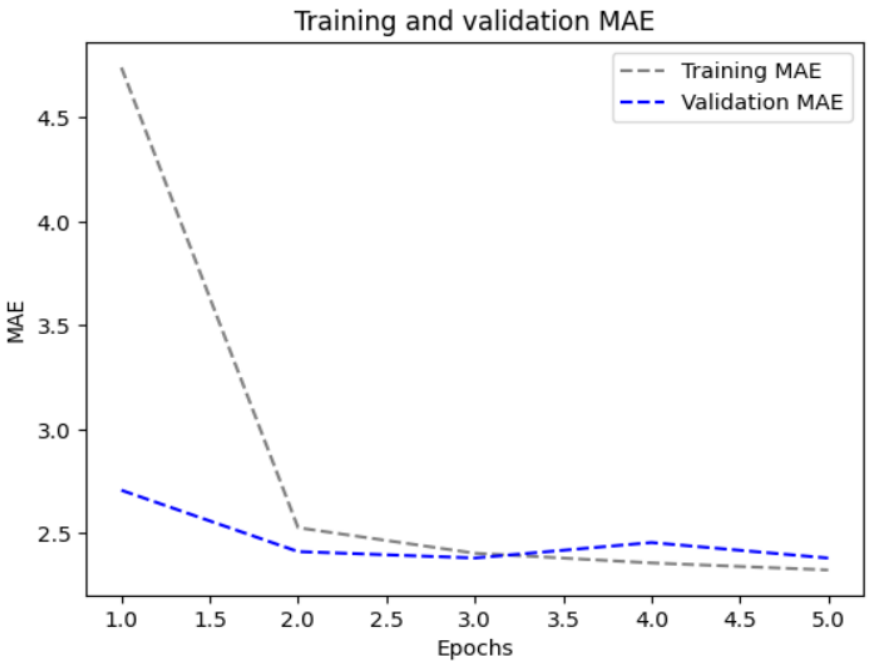
A Simple RNN

Recurrent neural networks are especially capable of embedding the history of time step information into today's decision-making process, enabling the identification of complicated associations and trends across sequential data. It is possible to describe sequences of different lengths because an RNN's internal state acts like a memory of previous inputs. Even though the simple RNN may preserve theoretically the data of all prior times, practical difficulties arise. This is what makes the training of deep networks difficult because of the problem known as the vanishing gradient problem. Also, it may be seen from the graph that the most basic RNN is the worst performer among all of these..

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

GRU

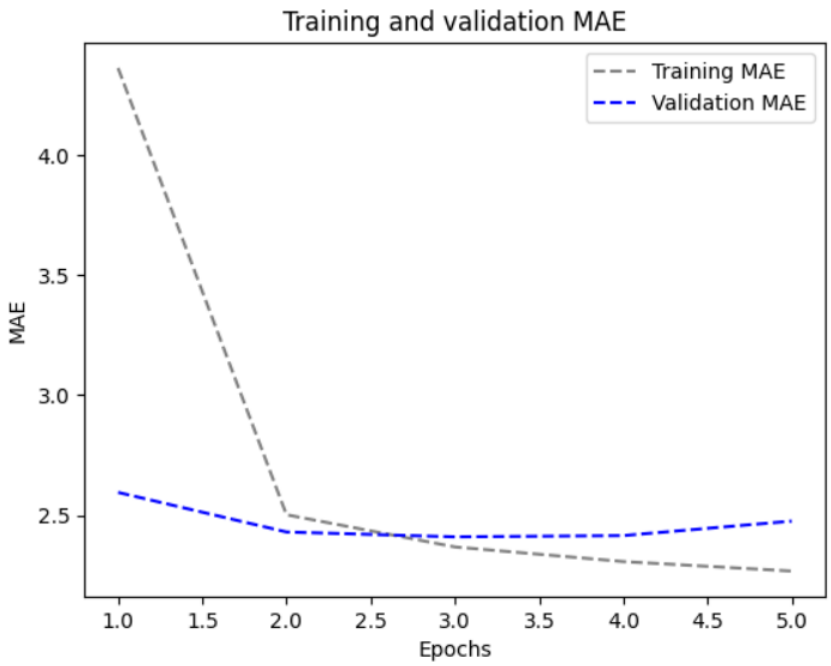
Instead of LSTM layers, we'll utilize Gated Recurrent Unit (GRU) layers. GRU and LSTM are quite similar; consider it an abridged, more straightforward form of the LSTM architecture.



The model MAE – 2.54 turns out to be the most efficient one because it is less expensive in terms of computational cost compared to other LSTM models. It captures long-range dependencies efficiently in the sequential data, too, as compared to other models.

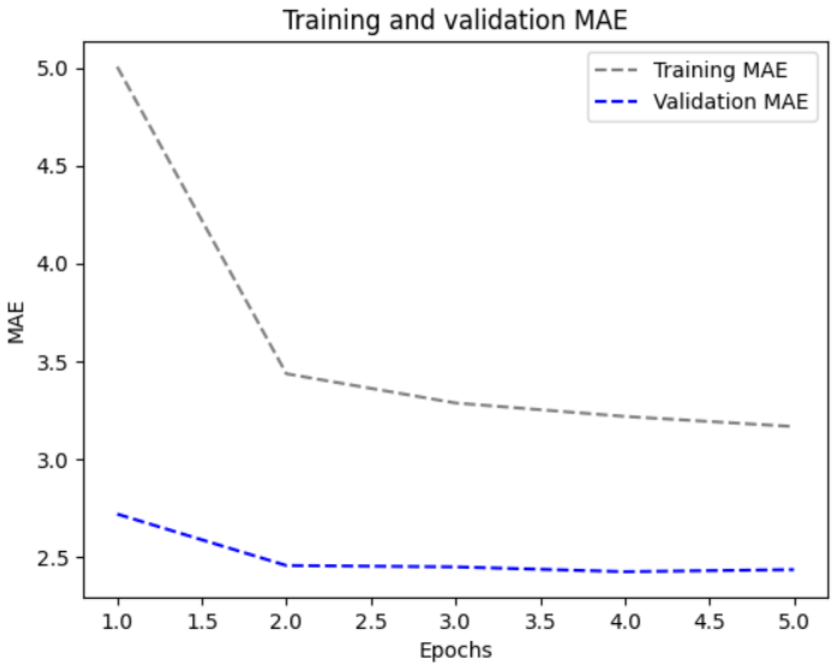
LSTM

Recurrent neural networks are a class of neural network topologies that were especially developed for this use application. One of the most popular methods is the Long ShortTerm Memory or LSTM layer. Let us try the LSTM layer first and we shall see how these models work 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.

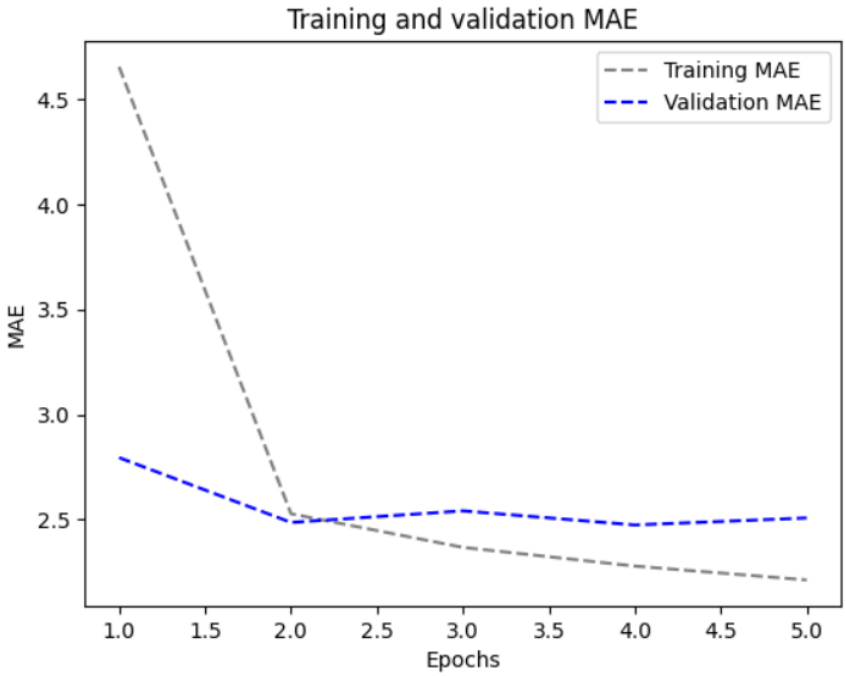
LSTM - dropout Regularization



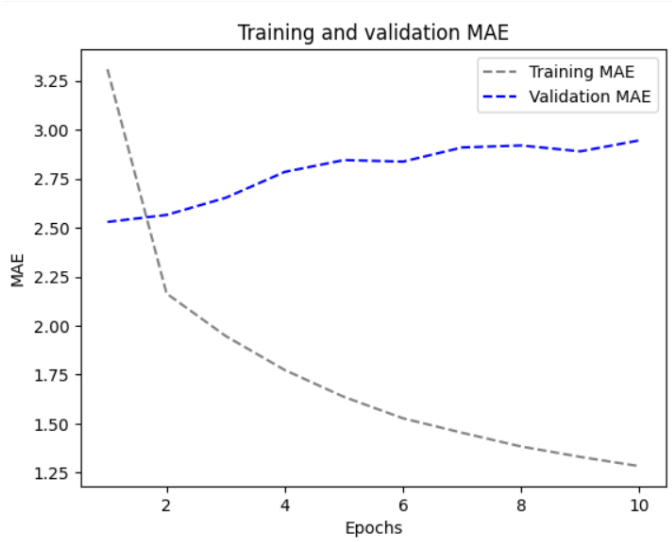
Achievement! The overfitting that occurred over the first 5 epochs has stopped. We attain a test MAE of 2.56 degrees and a validation MAE as low as 2.36 degrees. Not too horrible.

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

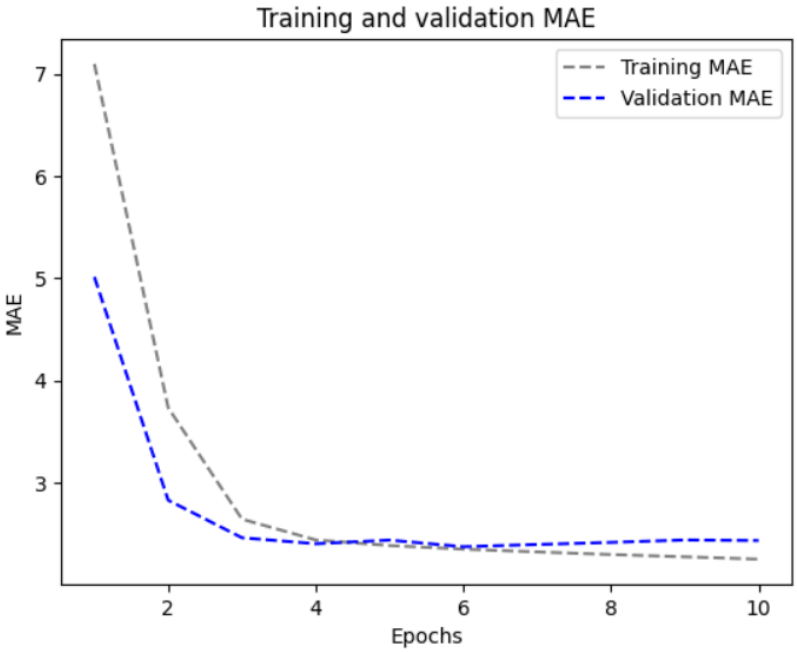
LSTM - Stacked setup with 16 units



LSTM - Stacked setup with 32 units

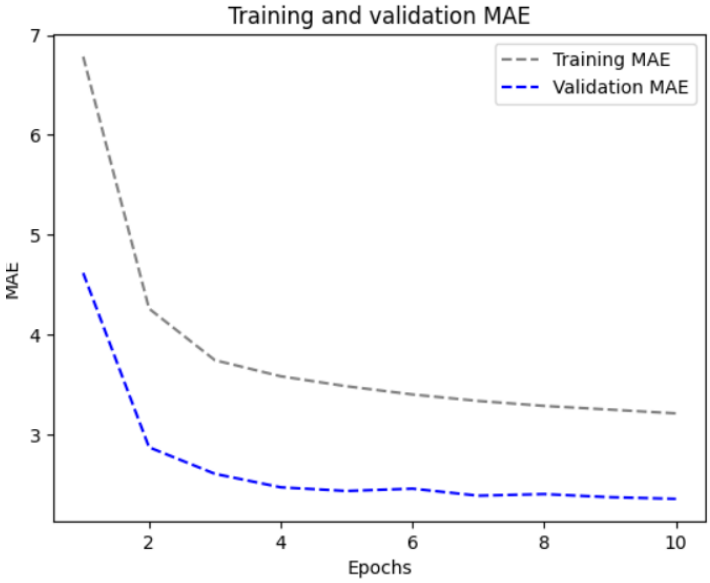


LSTM - Stacked setup with 8 units

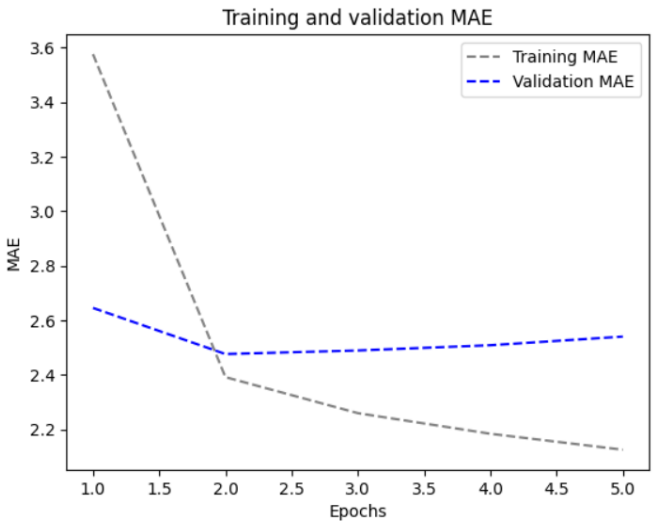


Among these variants, the 8-unit configuration was able to perform best and yield the MAE score of 2.58.
I also tried methods such as recurrent dropout to avoid overfitting and the use of bidirectional data, which lowers MAE values because information is given in another way to the recurrent network. Such enhancements resulted in roughly equal MAE values among them for all models, and most surprisingly, these values were shared below those of the common-sense model. What is even more striking-the same thing is confirmed by the graph of MAE assessment.

LSTM - dropout-regularized, stacked model

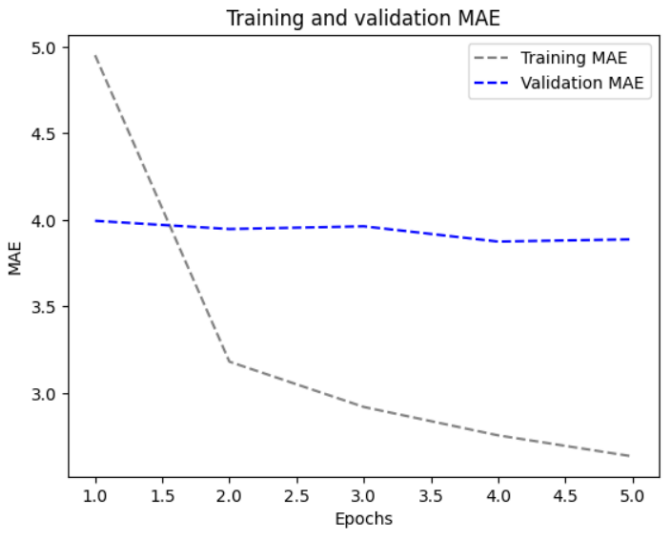


Bidirectional LSTM

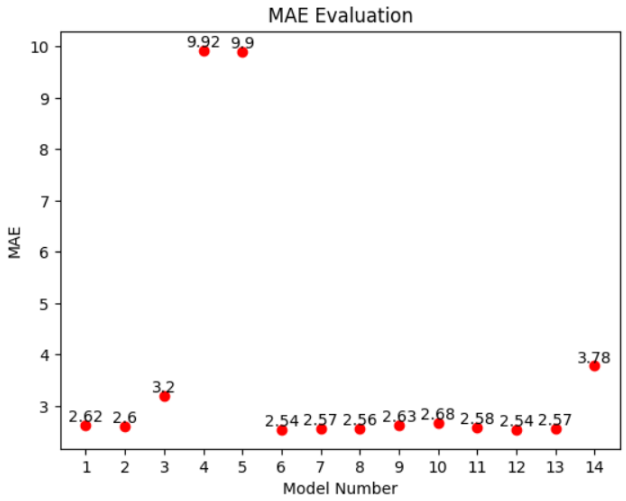


1D Convnets and LSTM together

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**

Generally, my results involve the use of LSTM and GRU-advanced RNN architectures. A combination with RNN and 1D convolution resulted in bad performance. In that respect, I consider that after some testing, GRU was found to be a better alternative when dealing with time series data although Bidirectional LSTM is still considered modern and widely used. Among the hyperparameters that are to be tuned for the GRU, towards its maximum performance, are the number of units in the stacked recurrent layers, the recurrent dropout rate, and whether to use bidirectional data.