ASSIGNMENT-3

Time-Series Data Report

Keerthi Tiyyagura

Rupesh Suragani

Introduction:

The purpose of this assignment is to apply RNNs (Recurrent Neural Networks) to time series data. We are demonstrating the utilization of temperature-forecasting task to underscore the distinctive characteristics that set timeseries data apart from the datasets you've previously encountered. Here, we are investigating several approaches for enhancing RNN models' performance in weather pattern predicting.

Goal of the project:

The goal of this project is to accomplish:

- How to apply RNNs to time series data
- How to improve performance of the neural network, especially when dealing with time-series data
- How to apply different deep learning layers to time-series data

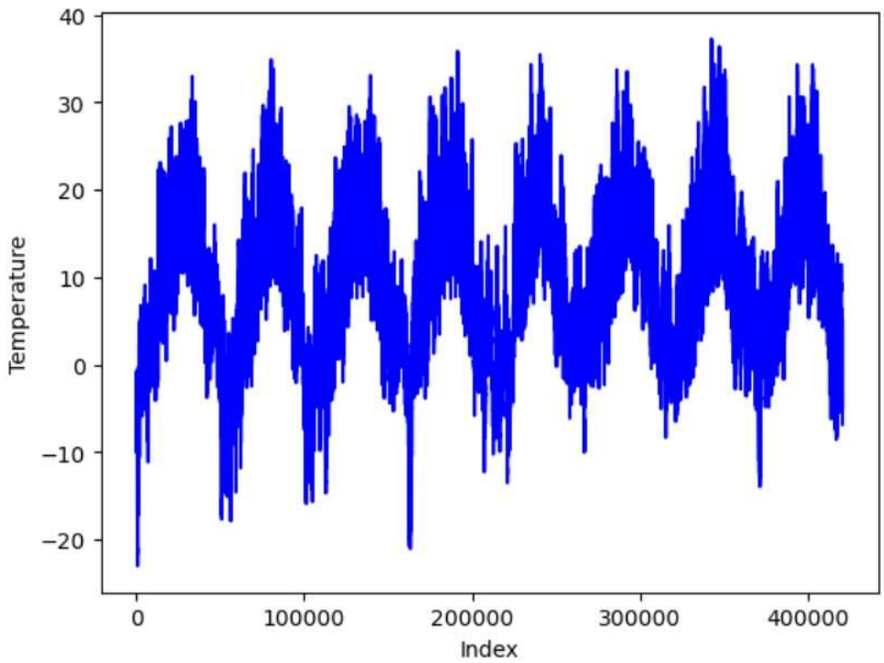
Methodology:

We have used Jupiter notebook for our assignment. For downloading weather forecast dataset, we have chosen AWS platform as a data source The dataset we have analyzed consists of time-series weather observations recorded at the weather station located at the Max Planck Institute for Biogeochemistry in Jena, Germany. These observations include 14 distinct variables such as temperature, pressure, humidity, wind direction, and more, recorded at 10-minute intervals over multiple years. Although the complete dataset spans back to 2003, the subset we'll be working with is restricted to the years 2009 through 2016.

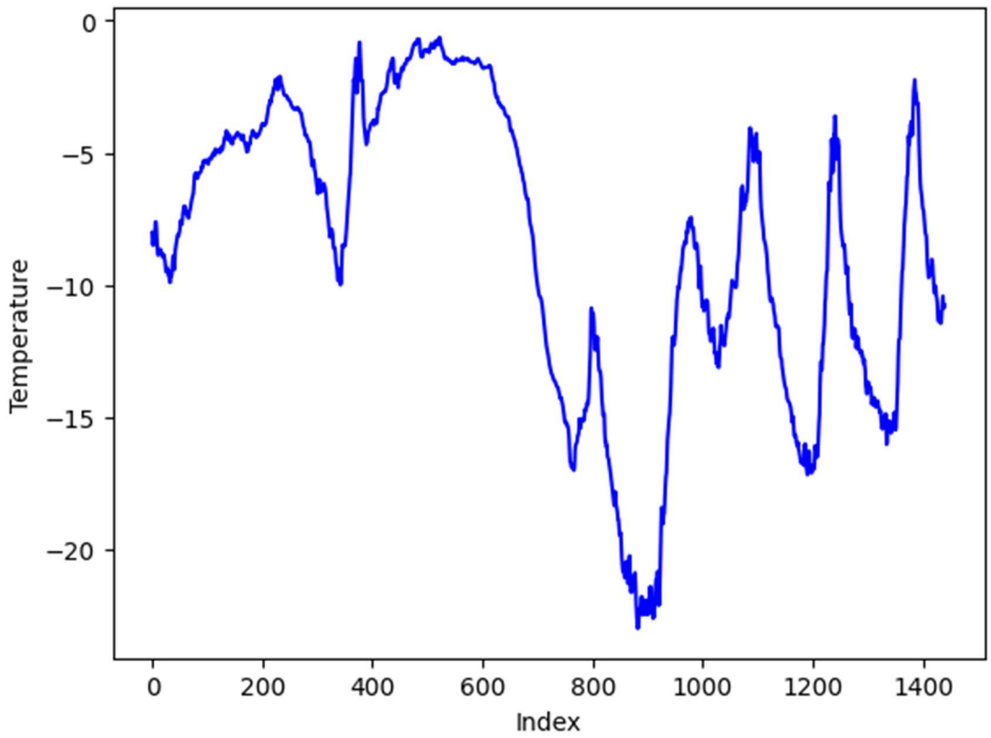
We have identified that the weather forecast csv data contains 420551 lines. These lines are converted into NumPy arrays of which one array is for the temperature (in degree

Celsius) and another one for the rest of the data so that we can use the features for predicting future temperatures.

The plot below illustrates the variation of temperature (measured in degrees Celsius) over time. It distinctly reveals the recurring yearly pattern in temperature observations, covering a time span of 8 years.



Below figure presents a focused depiction of the initial 10 days of temperature data. Given the data's recording frequency of every 10 minutes, each day encompasses 144 data points (24 hours * 6 data points per hour).

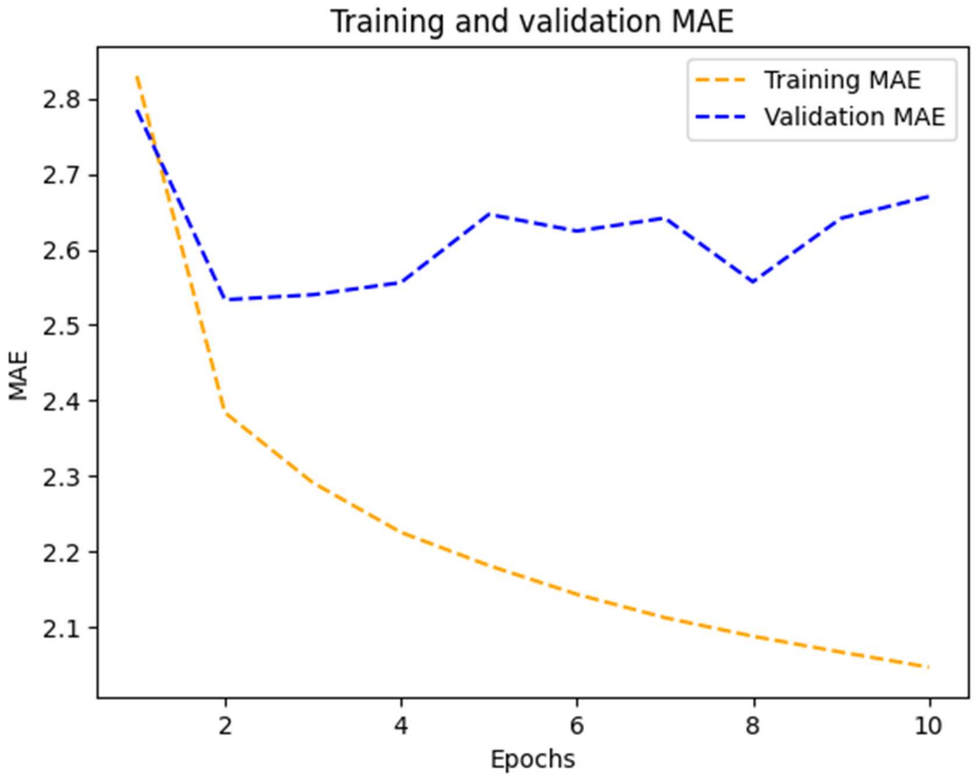


When attempting to predict the average temperature over the next month based on several months of past data, the task appears relatively straightforward due to the consistent yearly patterns present in the dataset. However, when focusing on shorter time scales, such as days, the temperature data appears more erratic and unpredictable. To assess the predictability of the time series on a daily scale, we will conduct experiments. Our approach involves dividing the dataset into three segments: 50% for training, 25% for validation, and 25% for testing. It's crucial to ensure that the validation and test data are more recent than the training data, as we aim to predict the future based on the past. This arrangement is essential because some problems become significantly simpler when the time axis is reversed.

Vectorization is unnecessary for this dataset since the data is already numerical. However, it's worth noting that each time series within the dataset operates on a different scale. We have normalized each timeseries independently so that they all take small values on a similar scale.

Here, we've employed a straightforward baseline approach where we predict the temperature for the next 24 hours to be the same as the current temperature. The resulting Mean Absolute Error (MAE) for the test dataset is 2.62 degrees Celsius, while

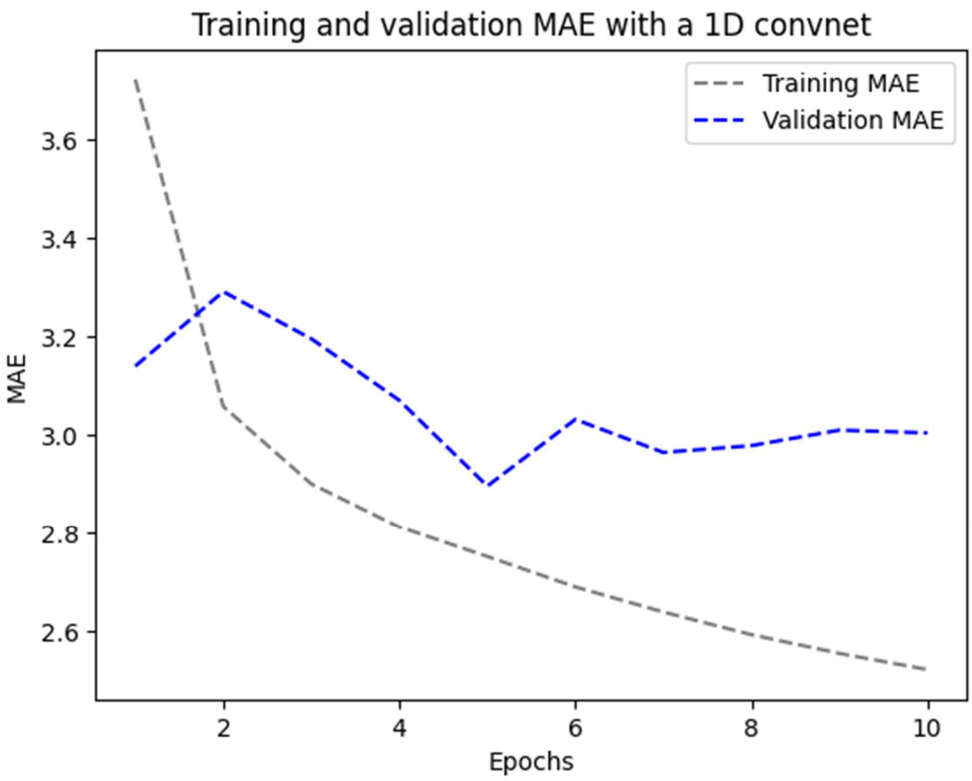
for the validation dataset, it's 2.44 degrees Celsius. Essentially, this means that assuming the temperature in the future remains constant at the current temperature would yield an average deviation of around two and a half degrees.



1D Convolution Model:

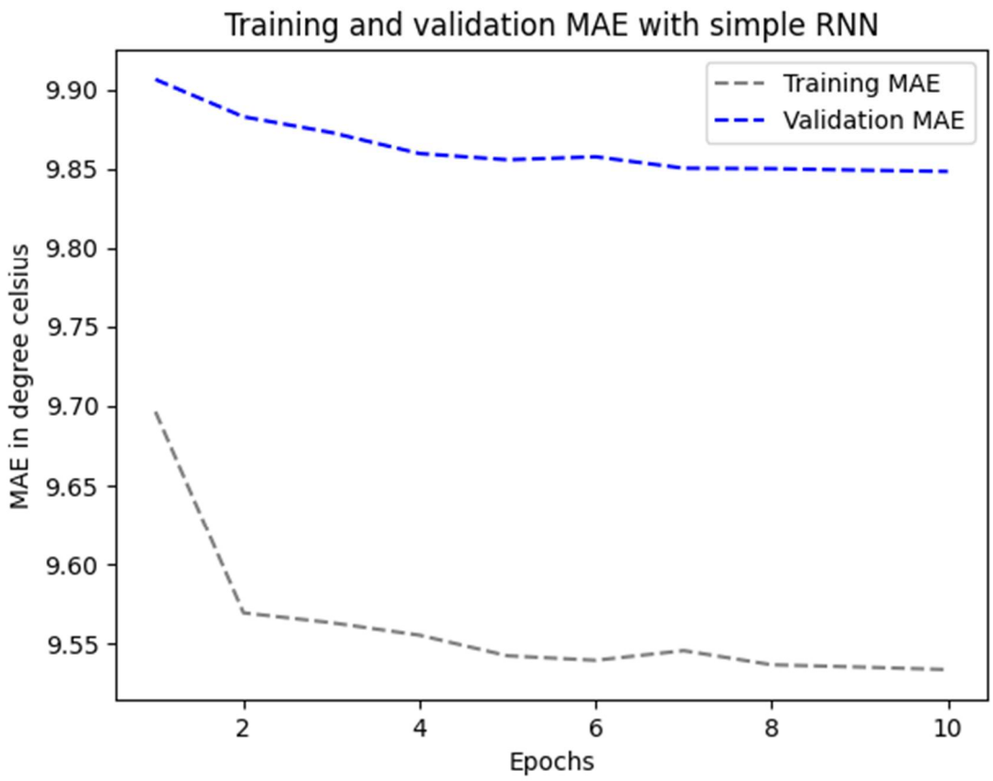
In addition to the familiar Conv2D and SeparableConv2D layers, there are also variations tailored for different dimensions: Conv1D and Conv3D. The Conv1D layer operates on 1D windows moving across input sequences, while Conv3D operates on cubic windows navigating through input volumes. Similar to 2D convolutions, these layers allow the construction of 1D convnets, which are particularly suitable for sequence data that exhibits translation invariance.

The performance of this model is notably inferior to that of the densely connected model. It achieves only a validation Mean Absolute Error (MAE) of approximately 3.10 degrees, which falls considerably short of the sensible baseline. This outcome highlights that not all meteorological data conforms to the assumption of translation invariance.



A Simple RNN:

Recurrent neural networks (RNNs) can recognize intricate relationships and patterns in sequential data because of their remarkable ability to integrate prior time step information into current decision-making processes. Sequences of varying lengths can be described since the internal state of an RNN serves as a memory for prior inputs. While a basic RNN may theoretically retain data from all previous times, practical challenges 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.

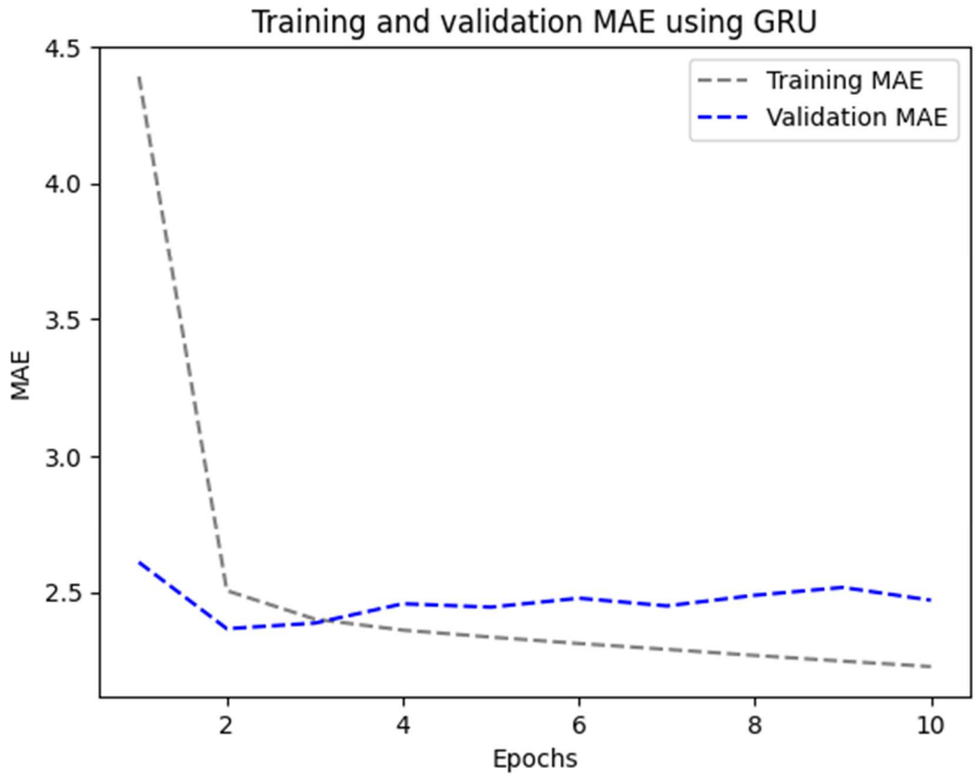


The test MAE for Simple RNN is 9.92-degree Celsius.

To overcome this problem, as a part of Kera's, we must create LSTM and GRU RNNs

GRU – Gated Recurrent Unit:

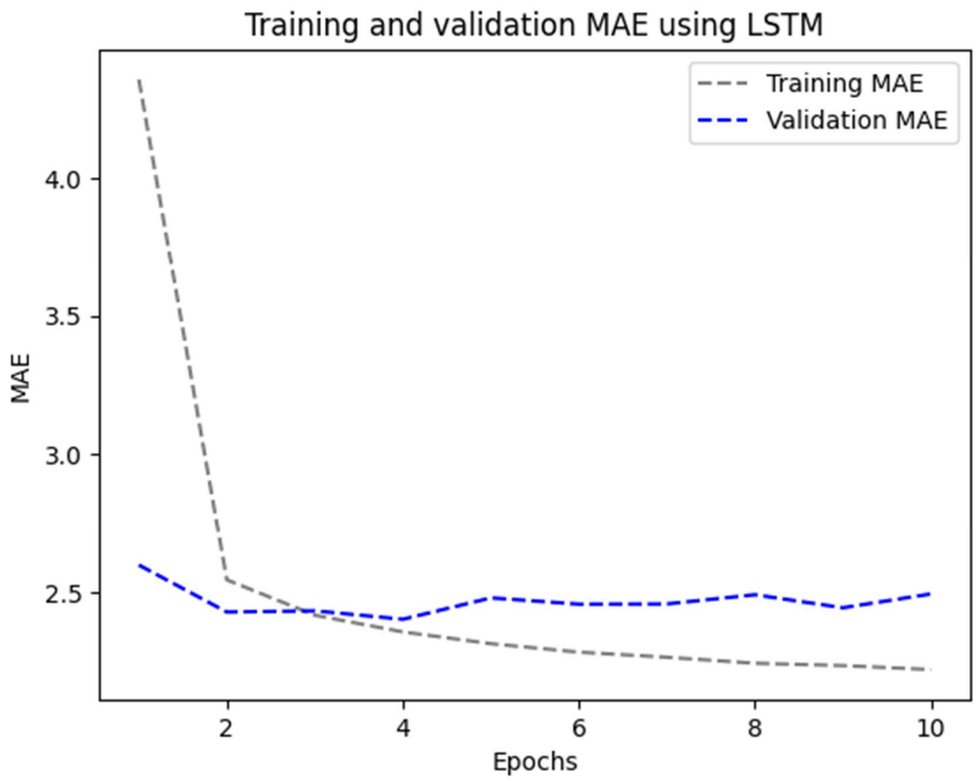
GRU, or Gated Recurrent Unit, is a type of recurrent neural network (RNN) architecture devised to mitigate the vanishing gradient problem in traditional RNNs. With fewer parameters than LSTM networks, GRUs employ reset and update gates to selectively retain or discard information over time, facilitating effective long-term dependency modeling.



We achieved Test MAE – 2.53 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 (Long Short-Term Memory):

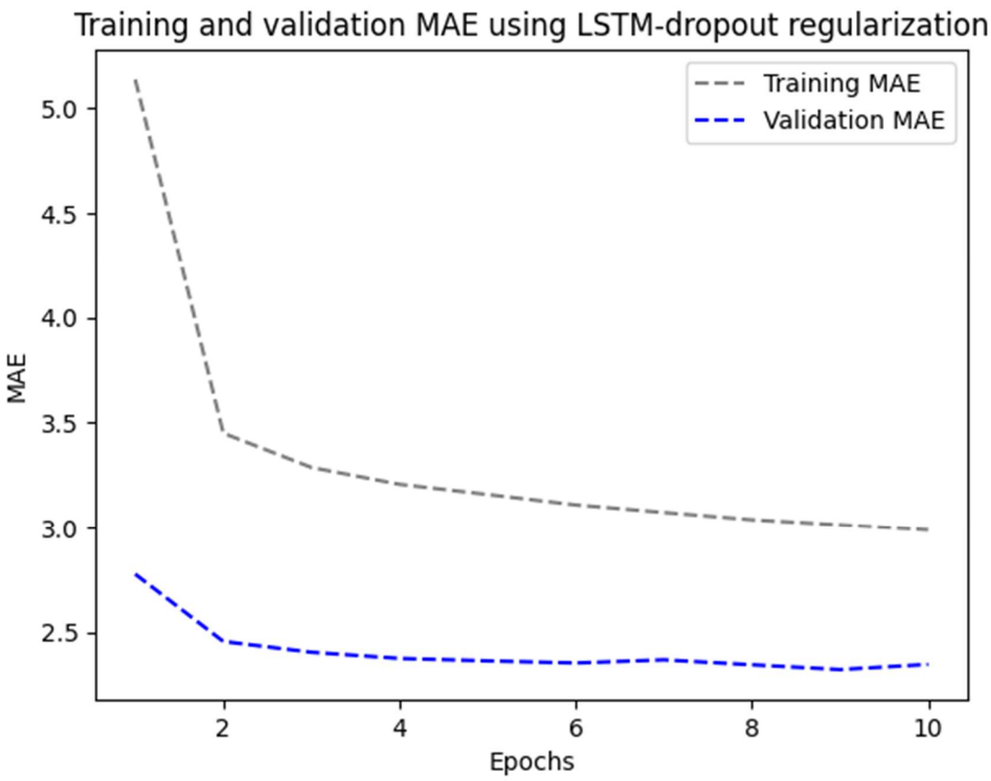
LSTM, or Long Short-Term Memory, is an RNN architecture designed to tackle the vanishing gradient problem and handle long-term dependencies. Using a memory cell and gates (input, forget, output), LSTM selectively retains or forgets information, making it effective for sequential data tasks.



The model with a Test MAE of 2.53 emerged as the most effective, proving to be less computationally intensive than LSTM models while still adeptly capturing long-range dependencies in sequential data compared to other models.

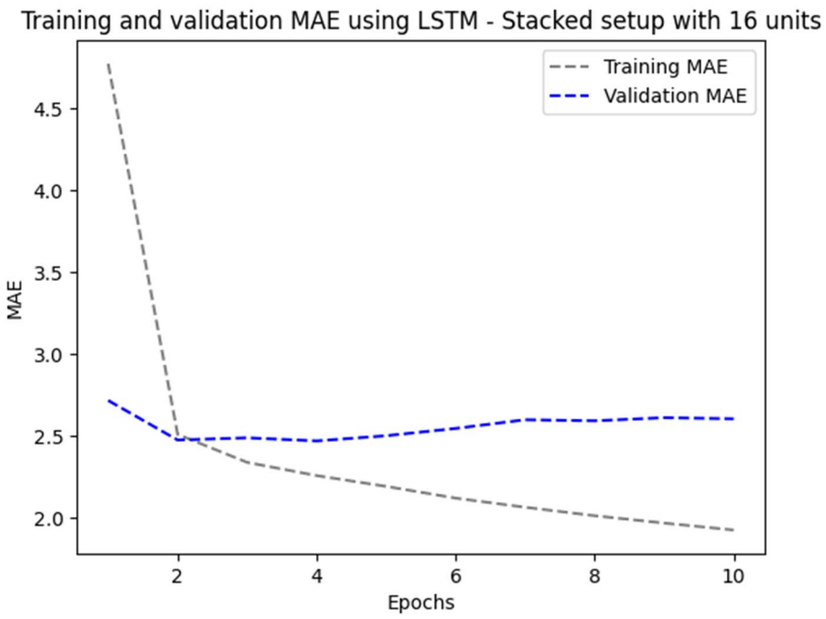
LSTM – dropout regularization:

LSTM dropout regularization applies dropout, a common neural network regularization technique, to connections between LSTM units. It randomly sets a fraction of input and recurrent connections to zero during training, aiding in preventing overfitting and enhancing model generalization.



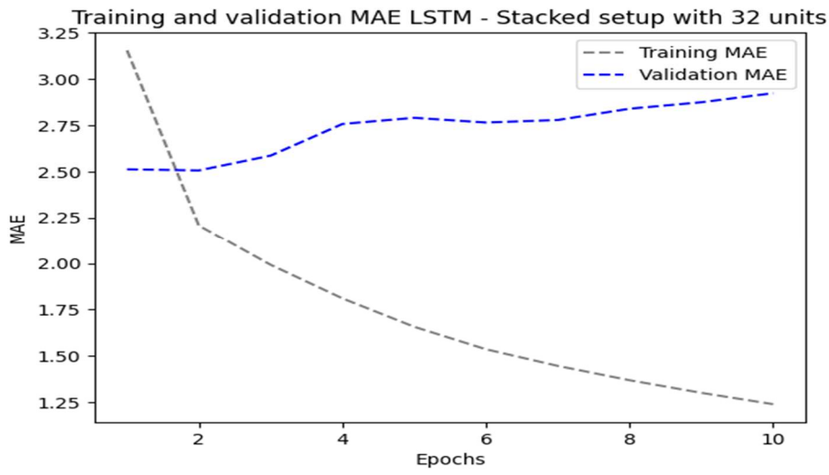
The overfitting observed during the initial 5 epochs has been mitigated. We achieved a test MAE of 2.56 degrees and a validation MAE as low as 2.36 degrees, which is satisfactory. I created six different LSTM models by varying the number of units inside the stacked recurrent layers to 8, 16, and 32 units.

LSTM - Stacked setup with 16 units



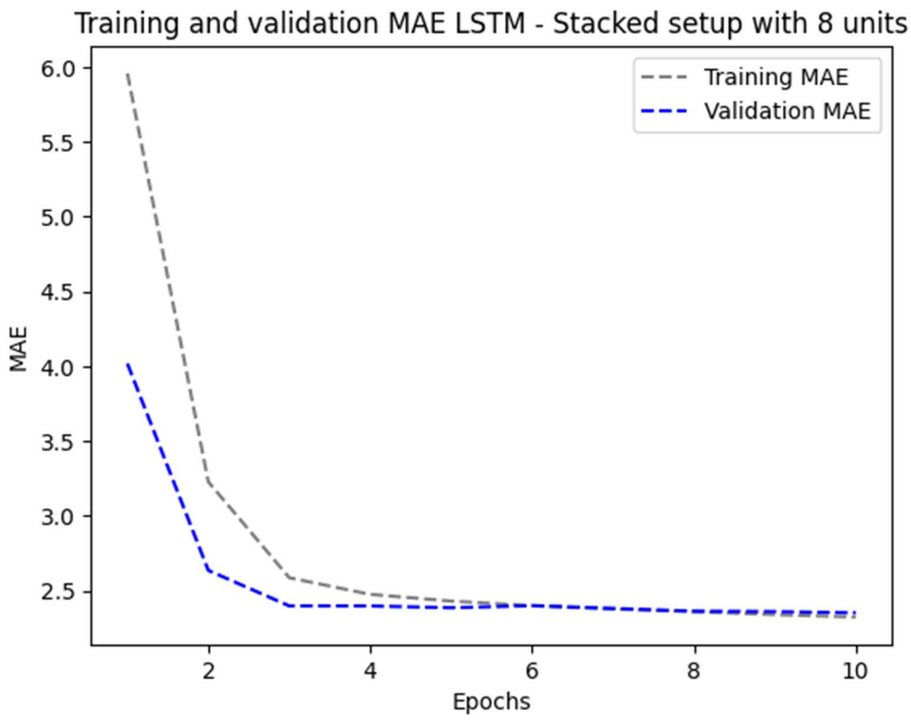
The test MAE for stacked setup with 16 units is 2.58 degree Celsius.

LSTM - stacked setup with 32 units



The Test MAE with a stacked setup with 32 units is 2.65

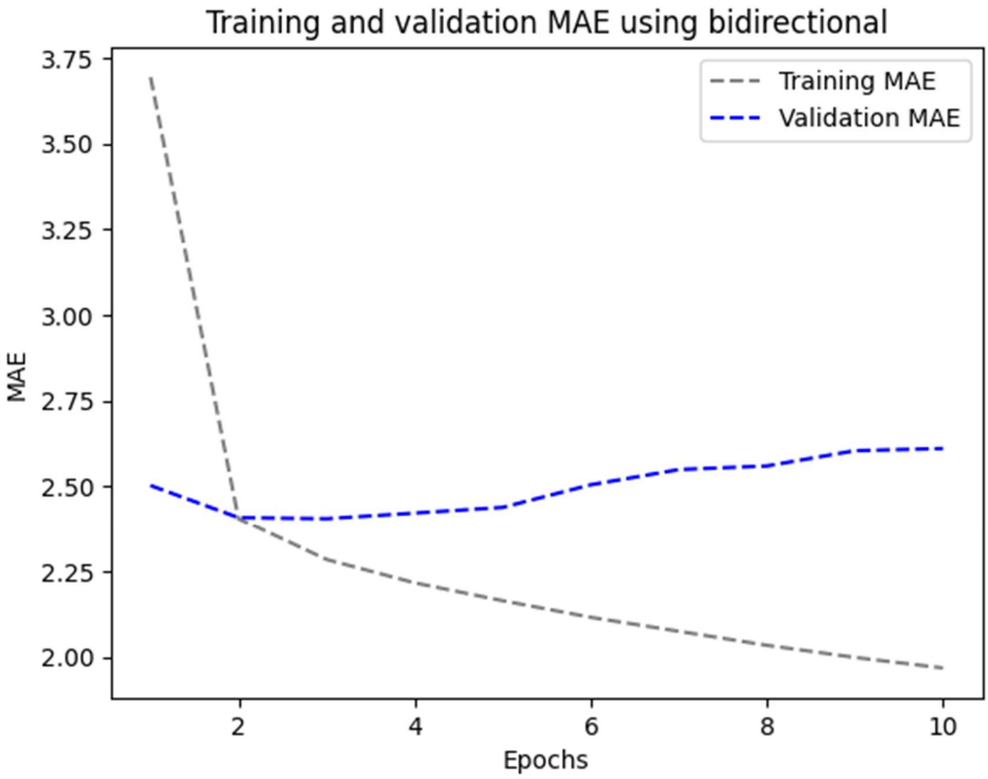
LSTM stacked setup with 8 units



The test MAE for stacked setup with 8 units is 2.53-degree Celsius.

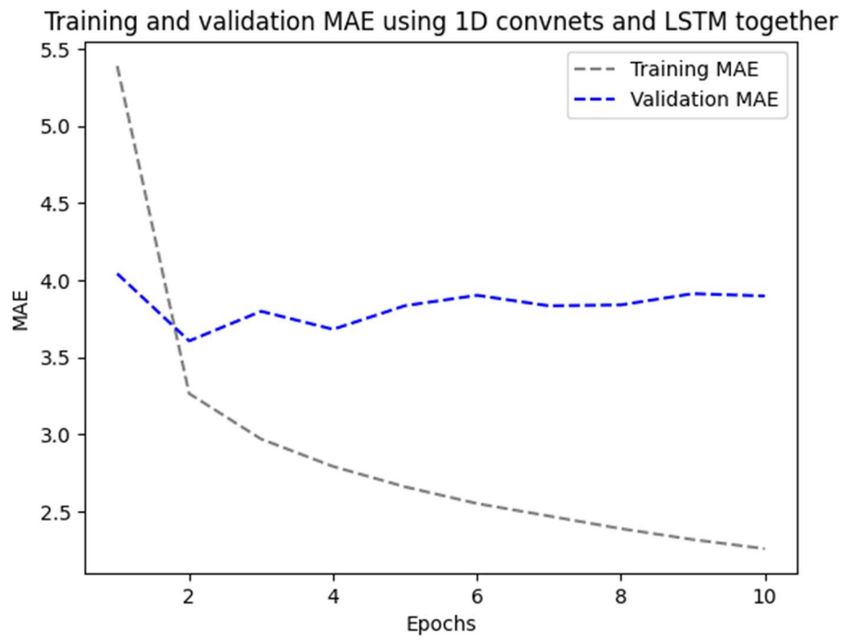
Bidirectional LSTM:

Bidirectional LSTM (BiLSTM) is a recurrent neural network architecture that processes input sequences in both forward and backward directions. This allows the network to consider both past and future information at each time step, improving its understanding of context and dependencies in sequential data.



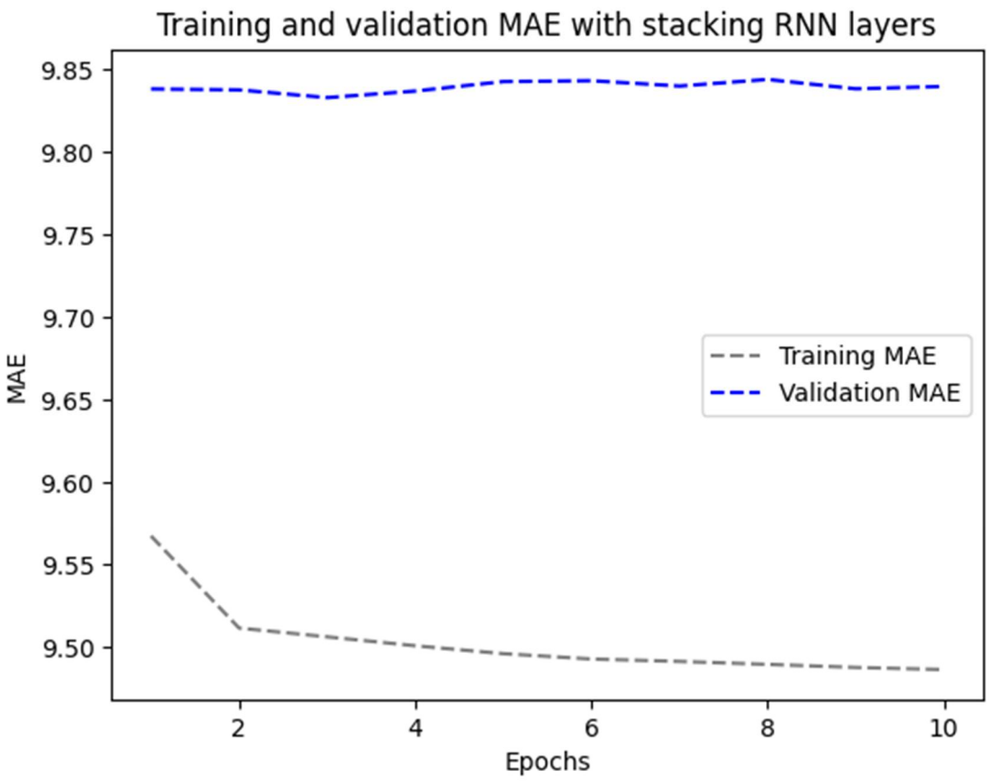
On evaluation of the test model the test MAE is 3.82 degree celsius.

1D convnets and LSTM together:



Stacked RNN:

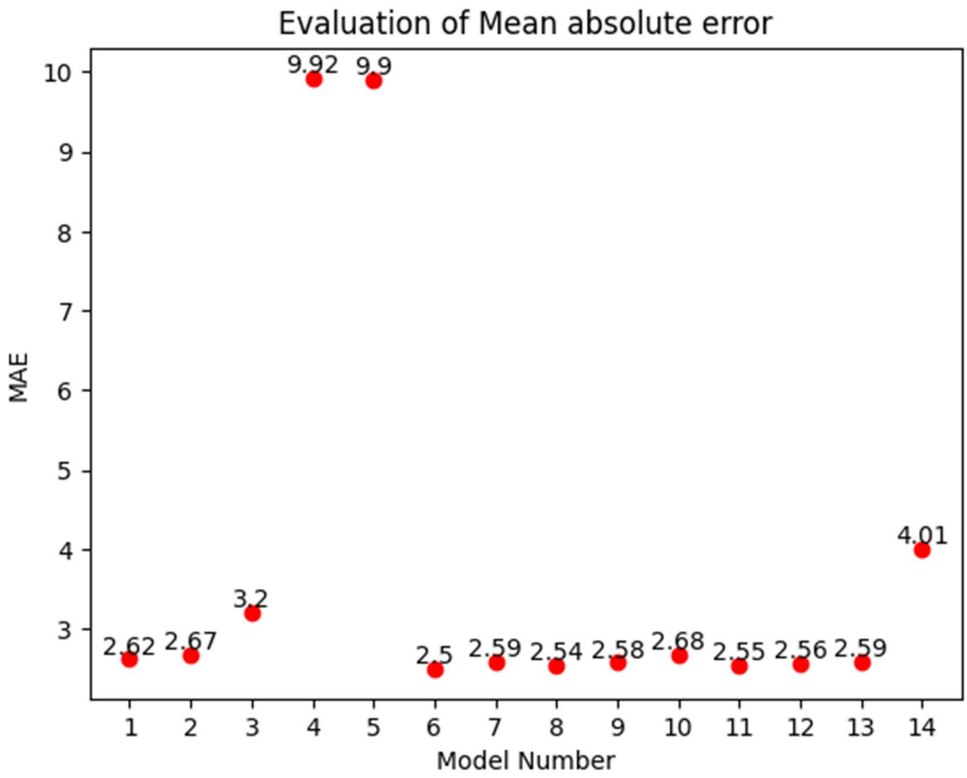
Stacked RNNs involve stacking multiple layers of recurrent units, enabling extraction of intricate temporal features from sequential data. Each layer receives input from the preceding one, enhancing the model's ability to capture long-term dependencies and hierarchical patterns. This architecture finds widespread application in natural language processing, time series prediction, and speech recognition tasks, where understanding sequential dependencies is paramount for precise predictions or classifications.



Throughout 10 training epochs, the model showed decreasing training loss and Mean Absolute Error (MAE). Validation metrics exhibited slight fluctuation, suggesting variability in model performance. The final test MAE provided insight into the model's performance on unseen data.

The final test MAE for stacked RNN model was 9.90, indicating the model's performance on unseen data. Overall, the model demonstrates modest performance, but there might be room for further improvement.

The overall evaluation of Mean absolute error (MAE):



The tabular format of all the MAE results are as follows:

S.No	Results : MODEL	Validation MAE	Test MAE
1.	Dense model	2.66	2.60
2.	1D convolutional Model	2.98	3.2
3.	Simple RNN	9.83	9.92
4.	Stacked Simple RNN	9.80	9.90
5.	GRU	2.43	2.54
6.	LSTM simple	2.39	2.57
7.	LSTM -dropout Regularization	2.36	2.56
8.	LSTM- Stacked 16 units	2.58	2.63
9.	LSTM – Stacked 32 units	2.94	2.68
10.	LSTM – Stacked 8 units	2.42	2.58
11.	LSTM – dropout Regularization, stacked model	2.36	2.57
12.	Bidirectional LSTM	2.45	2.57
13.	1D convolutional and LSTM	3.84	3.78

Conclusion:

In conclusion, my findings indicate that employing LSTM and GRU, advanced RNN architectures, yields better results compared to combining RNN with 1D convolution, which proved ineffective. Through experimentation, I've determined that GRU is particularly effective for handling time series data, although Bidirectional LSTM remains a popular choice. Key hyperparameters to optimize GRU performance include adjusting the number of units in stacked recurrent layers, the recurrent dropout rate, and considering bidirectional data usage.