ASSIGNMENT-3

Time - Series Data

PROBLEM STATEMENT

Design and implement a Recurrent Neural Network (RNN) model to predict the future values of a time-series dataset. The dataset consists of historical observations collected at regular intervals, and the objective is to train the RNN model to learn patterns and relationships within the time-series data in order to accurately forecast future values.

INTRODUCTION

Forecasting future temperatures using past data is difficult, especially with time series datasets. Regular networks struggle with this, but recurrent neural networks (RNNs) are great for it. We are using a weather dataset from Germany that covers 2009 to 2016. It includes lots of measurements like temperature, pressure, and humidity, taken every 10 minutes. With RNNs, we are trying to predict temperatures accurately. This shows how well RNNs can handle time series data like this.

TECHNIQUES

Preprocessing

Before feeding the data into a neural network, it requires preprocessing. Since each timeseries in the dataset is on a different scale, normalization is crucial to ensure that they all have values on a similar scale. We compute the mean and standard deviation on a fraction of the data (the first 210,225 timesteps) and normalize each timeseries independently.

Data Preparation

We need to structure the data into a format suitable for training a neural network. To achieve this, we create a Dataset object that yields batches of data from the past five days, along with the target temperature 24 hours in the future. We utilize the built-in utility **"timeseries_dataset_from_array ()"** in Keras to efficiently generate samples on the fly, avoiding the need to allocate memory for every sample explicitly.

Dataset Parameters

Three datasets are instantiated for training, validation, and testing, respectively. We specify parameters such as sampling_rate (1 data point per hour), sequence_length (5 days or 120 hours), and delay (target temperature 24 hours after the end of each sequence).

Baseline Approach

Before delving into complex deep learning models, it's prudent to establish a baseline using a simple, common-sense approach. In this case, we assume that the temperature 24 hours from now will be equal to the current temperature. This baseline achieves a validation Mean Absolute Error (MAE) of 2.44 degrees Celsius and a test MAE of 2.62 degrees Celsius.

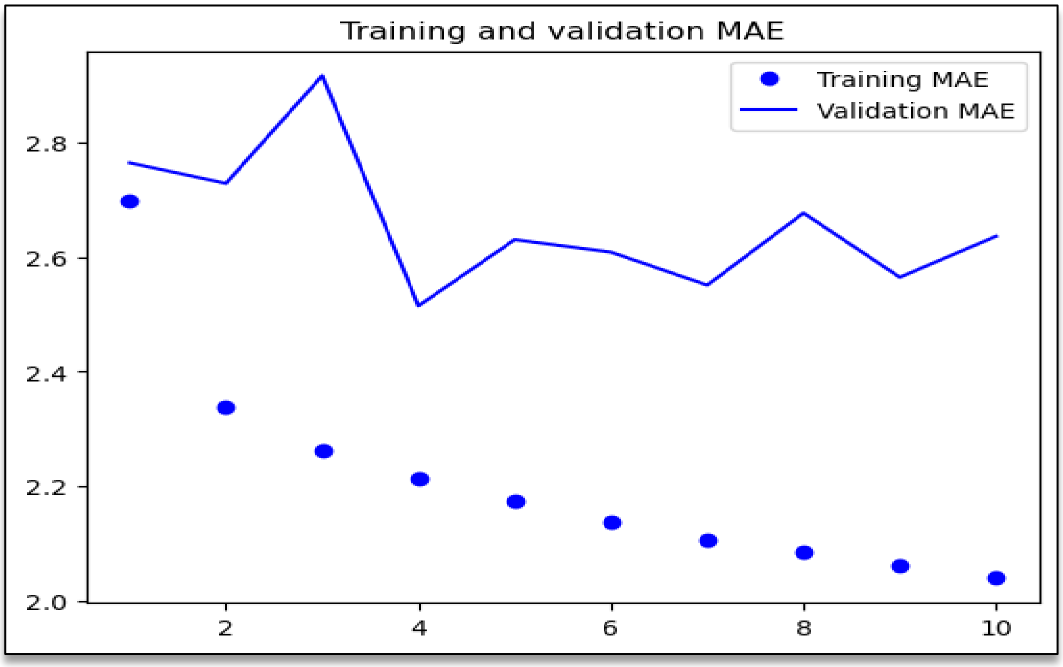
Deep Learning Techniques

With the baseline established, we aim to improve temperature prediction accuracy using deep learning techniques. Recurrent Neural Networks (RNNs) are particularly well-suited for time series forecasting tasks due to their ability to retain information over time. We'll explore architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which are specialized RNN variants designed to address the vanishing gradient problem and capture long-term dependencies in the data.

RESULTS

Basic machine learning model: Dense Layer

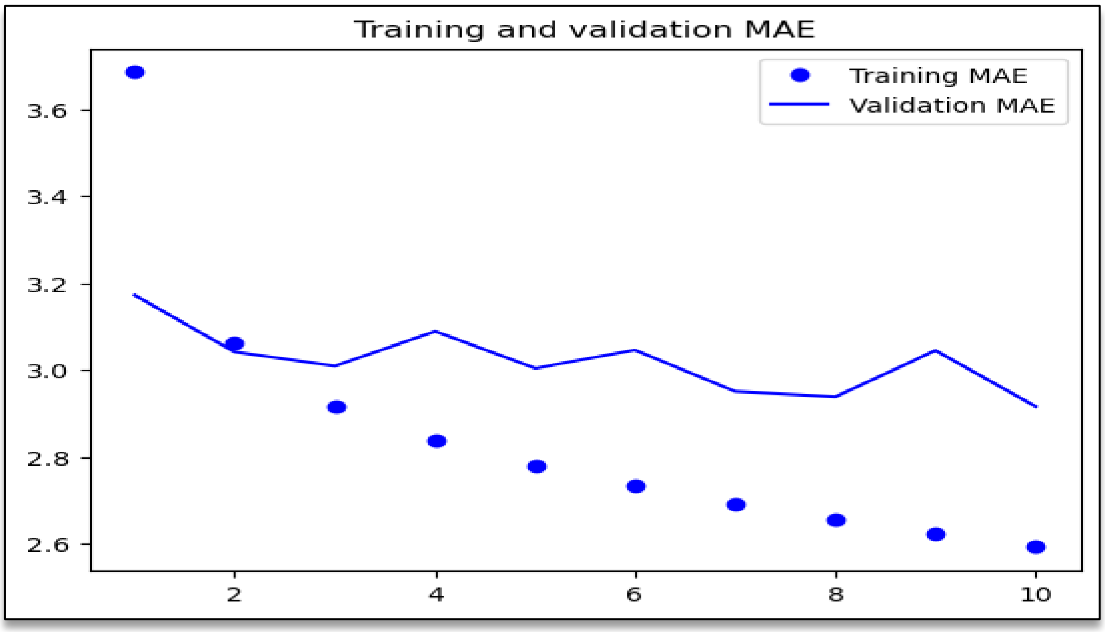
Starting with common-sense baselines and simple models like small, densely connected networks before more complex ones such as RNNs ensures genuine improvements. This approach entails utilizing a basic fully connected model with flattening and Dense layers, often without activation functions for regression. Key metrics like MSE and MAE are monitored. Despite simple solutions existing within the model's hypothesis space, finding them via gradient descent is challenging due to the expansive search space. This highlights the crucial role of feature engineering and guiding the model with pertinent architecture priors for successful problem-solving.



The model achieves a reported validation MAE of 2.63 and test MAE of 2.59.

1D Convolutional Model

Trying a convolutional model for our temperature forecasting, with Conv1D layers, seemed promising due to its ability to extract patterns from sequences. However, it performed worse than expected, with a validation MAE of around 2.9 degrees. Two main issues arose: Firstly, weather data doesn't fully adhere to the assumption of translation invariance, as properties vary throughout the day. Secondly, the order of data matters significantly, with recent data being more predictive. Unfortunately, the Conv1D model couldn't capture this, especially with max pooling layers disrupting order information. Hence, the model failed to outperform simpler approaches due to these limitations.



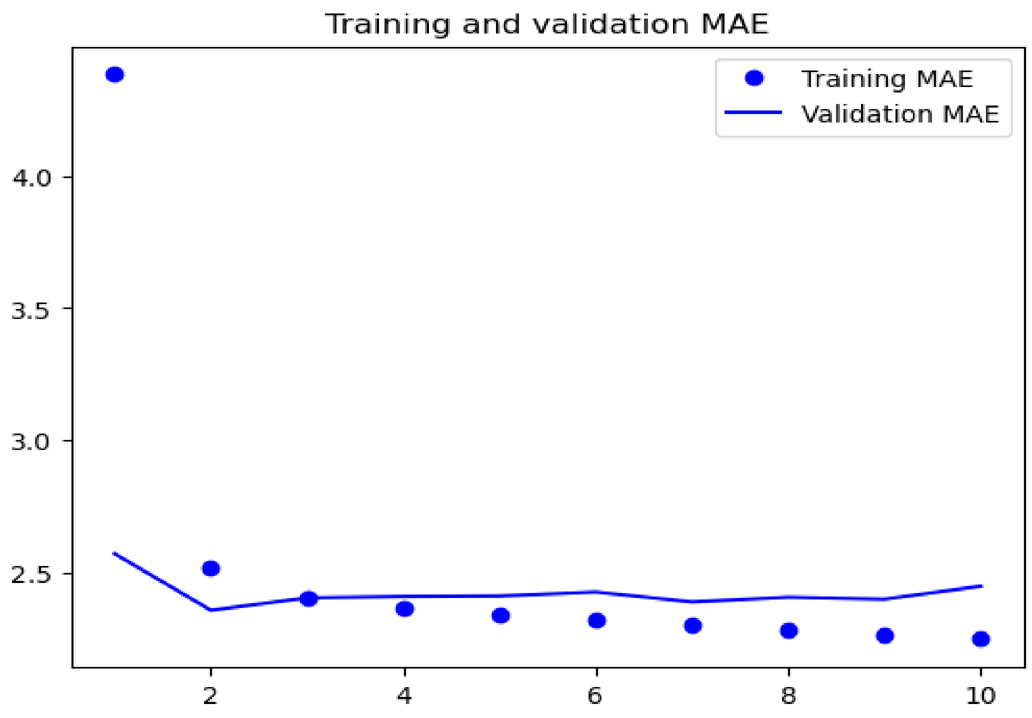
The model achieves a reported validation MAE of 2.91 and test MAE of 3.01.

A Simple RNN

RNNs excel at capturing temporal dependencies in data by retaining information from past time steps, facilitating complex pattern recognition. Despite the potential to store information from any historical point, practical challenges like the vanishing gradient issue hinder deep RNN training. Moreover, basic RNNs often underperform, as illustrated in the graph. To address these limitations, we turn to GRUs and LSTMs.

GRU's

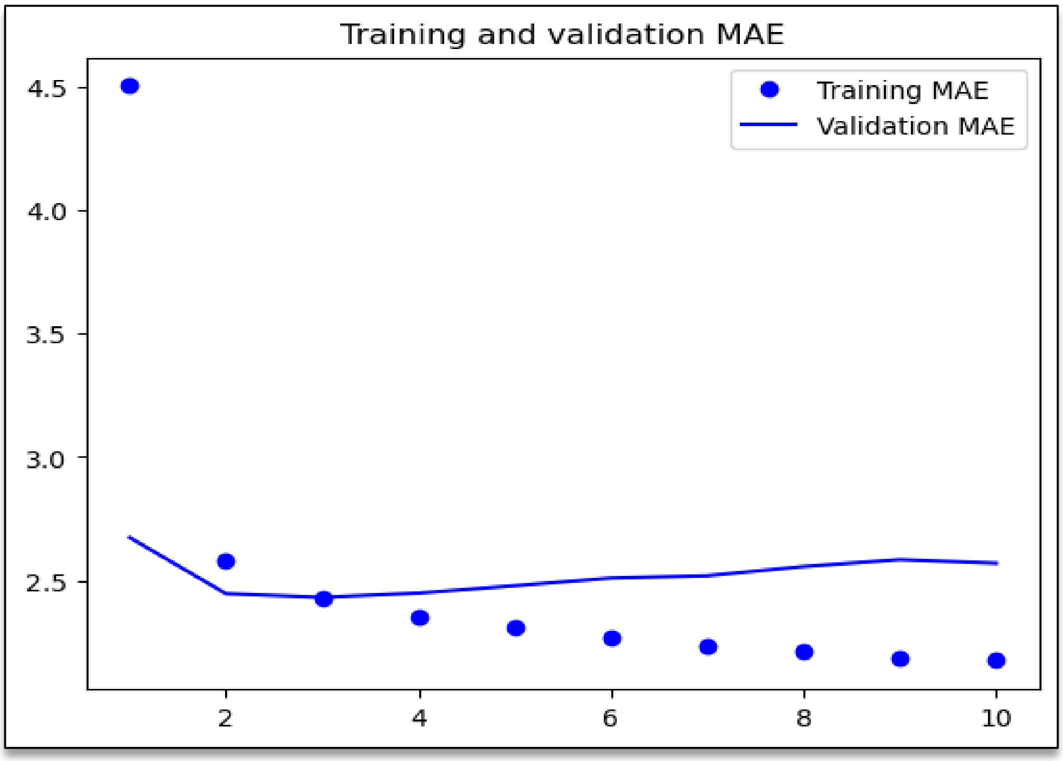
We opt for Gated Recurrent Units (GRUs) instead of LSTMs. GRUs offer a simplified version of LSTM architecture, maintaining its effectiveness while being more streamlined.



The model achieves a reported validation MAE of 2.44 and test MAE of 2.53.

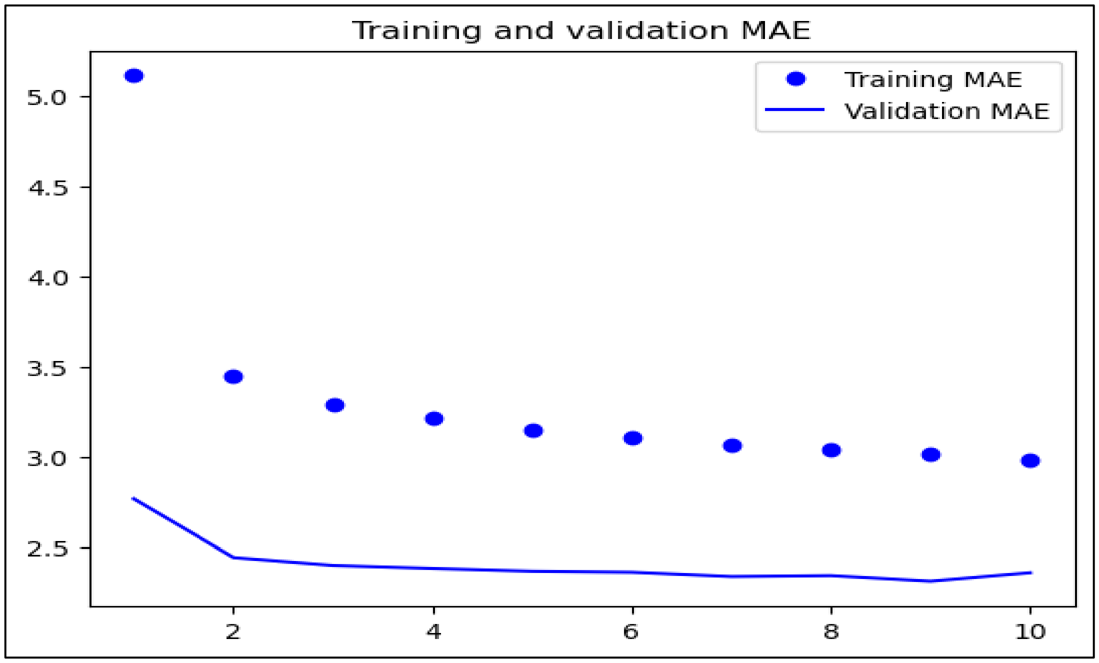
LSTM Models

A type of neural network architecture created especially for sequential data processing tasks is offered by recurrent neural networks (RNNs). The Long Short-Term Memory (LSTM) class is the most well-liked among them. To learn about the functions of the LSTM layer, we will first explore it.



Our results showed a 2.61-degree test MAE and a 2.57-degree validation MAE. The LSTM-based model performed better than the traditional baseline, demonstrating the value of machine learning in this endeavor.

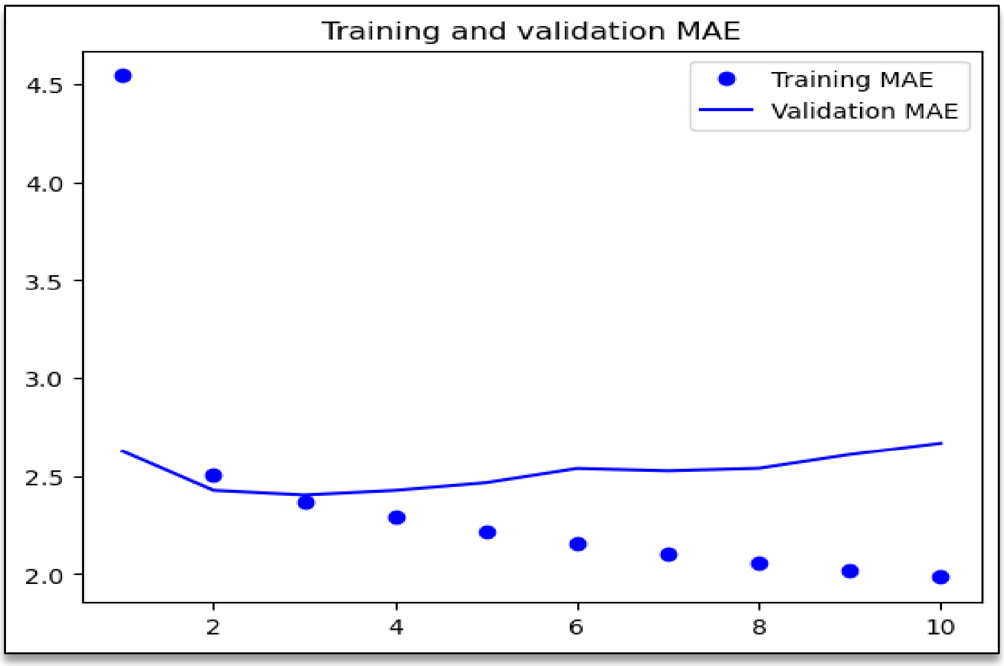
LSTM- Dropout Regularization



The model achieves a reported validation MAE of 2.36 and test MAE of 2.52.

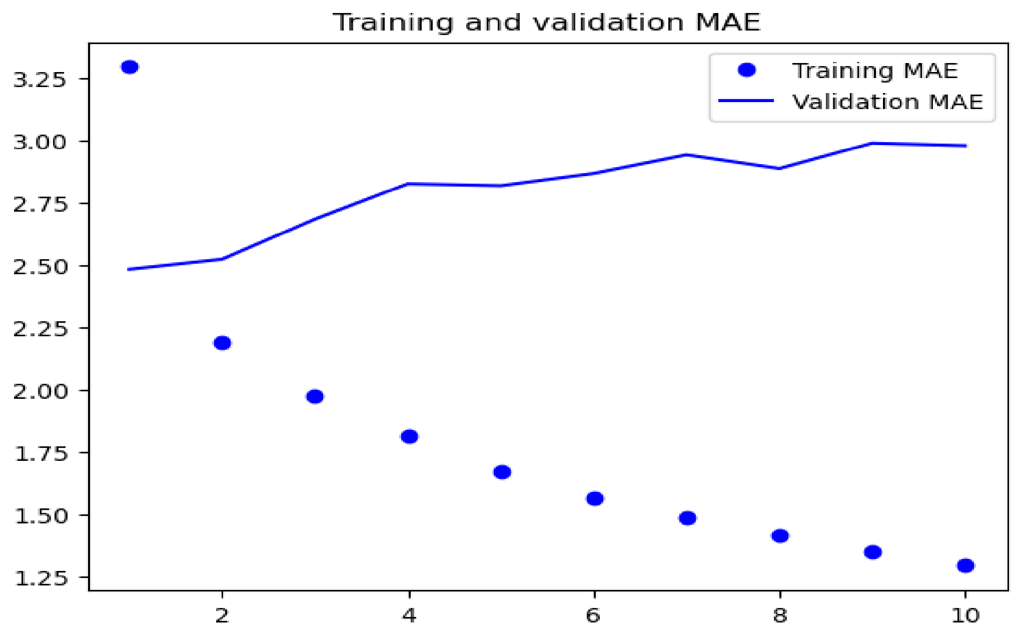
We built six different LSTM models with 8, 16, and 32 units as the number of units that vary inside the stacked repetition layers.

LSTM stacked with 16 units



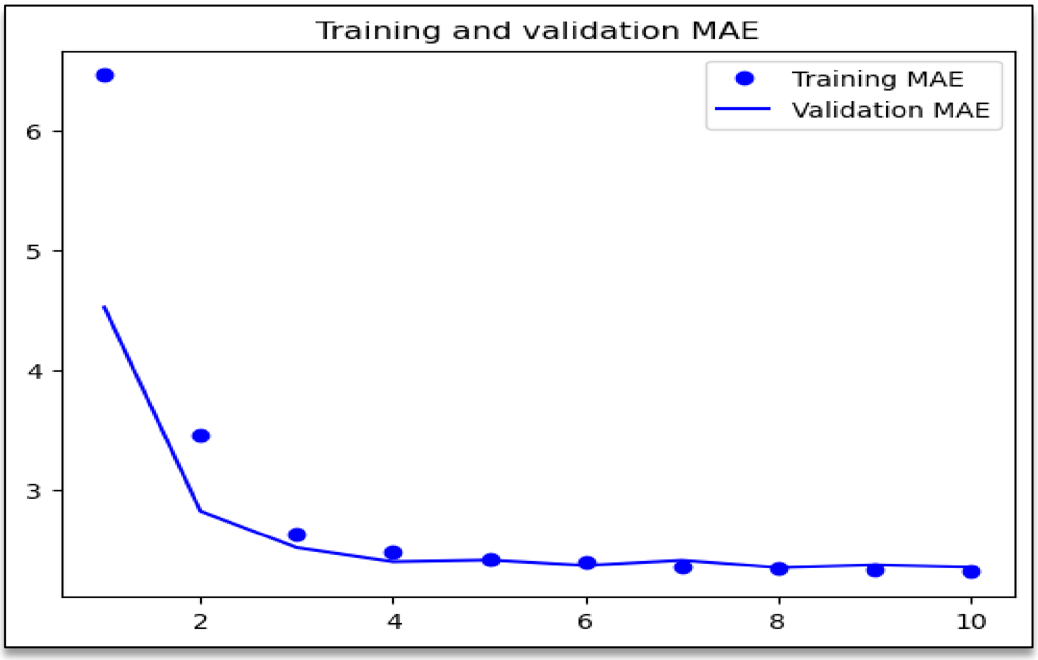
The model achieves a reported validation MAE of 2.66 and test MAE of 2.56.

LSTM stacked with 32 units



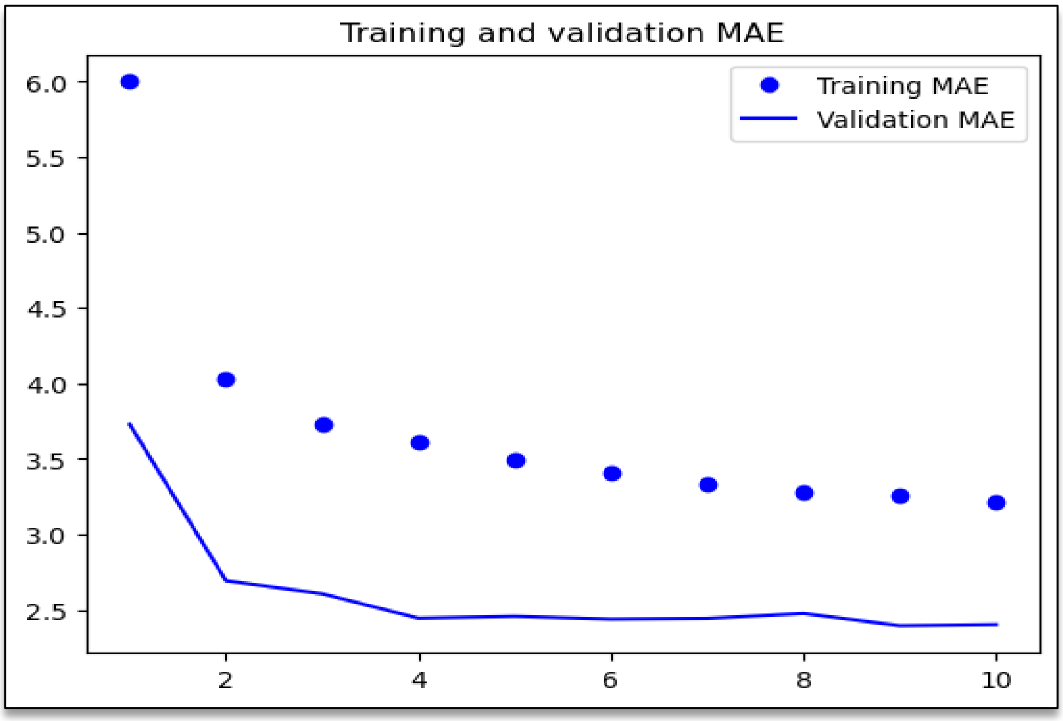
The model achieves a reported validation MAE of 2.98 and test MAE of 2.64.

LSTM stacked with 8 units



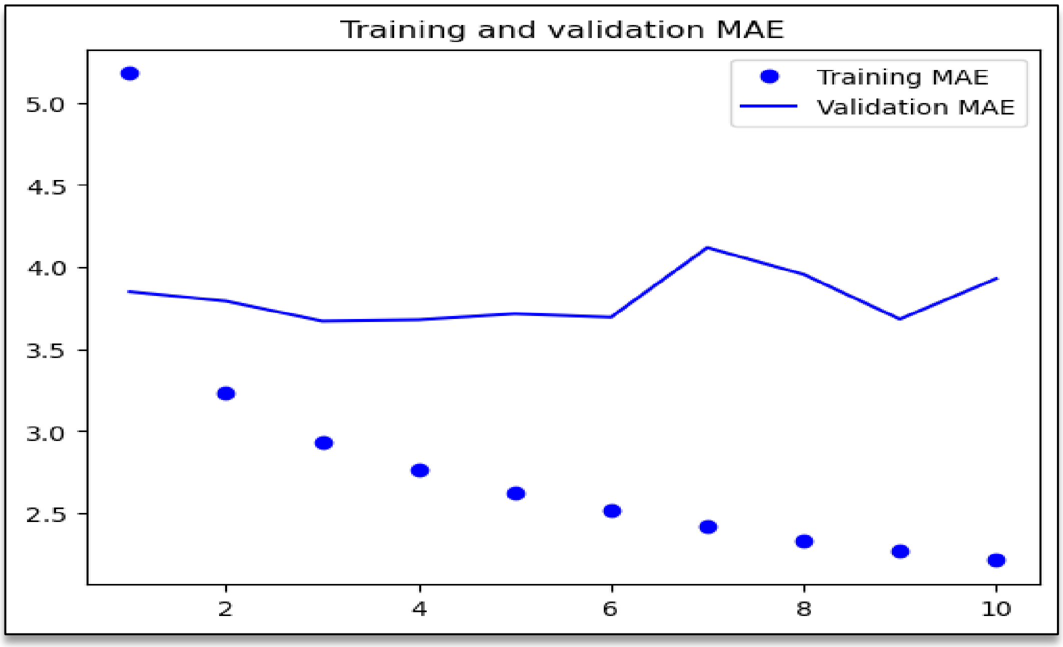
The model achieves a reported validation MAE of 2.35 and test MAE of 2.52.

LSTM - dropout-regularized, stacked model



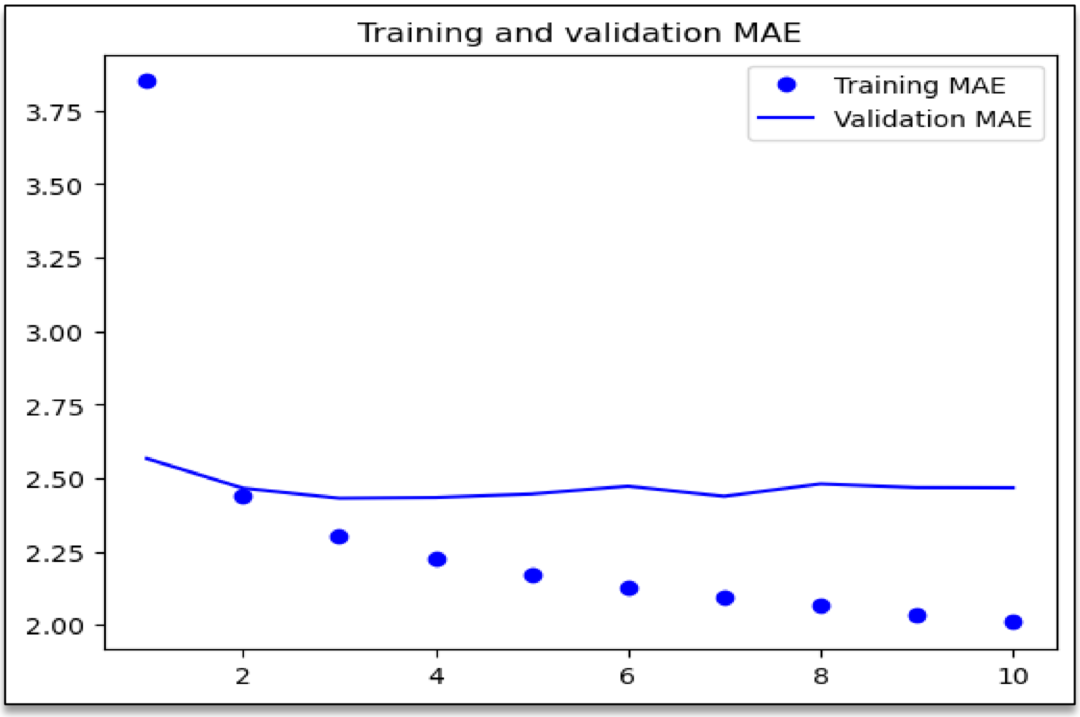
The model achieves a reported validation MAE of 2.40 and test MAE of 2.57.

1D Convnets and LSTM together



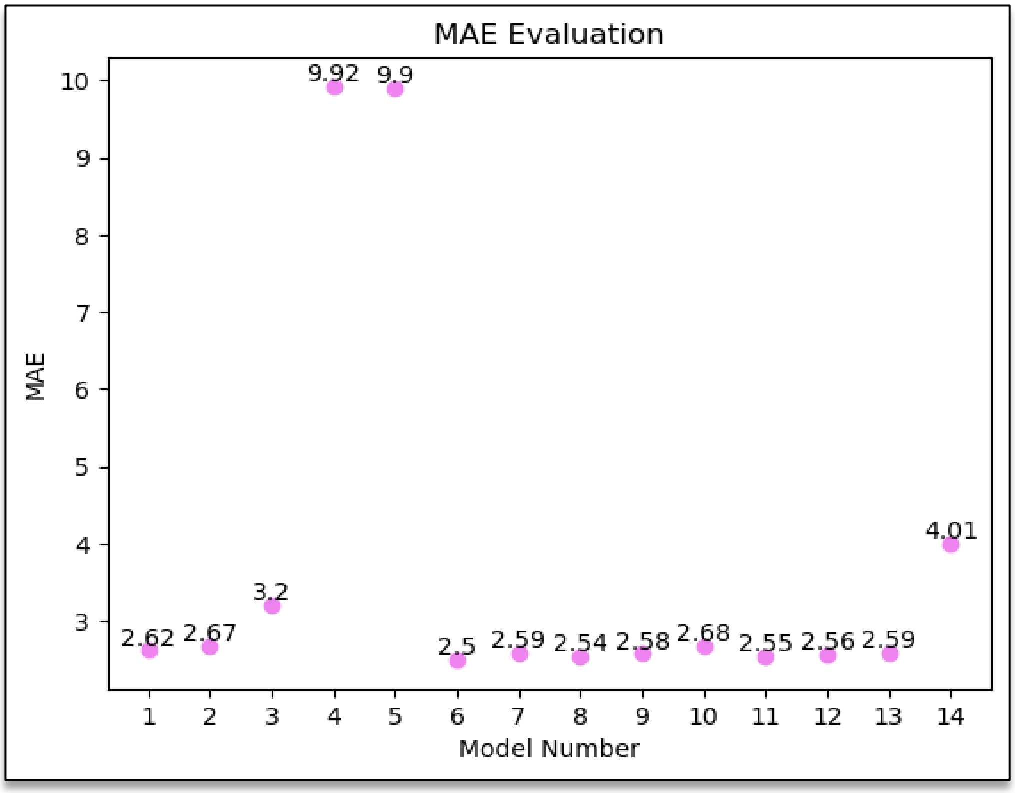
The model achieves a reported validation MAE of 3.92 and test MAE of 3.79 .

LSTM: Bidirectional



The model achieves a reported validation MAE of 2.46 and test MAE of 2.60.

The Complete Model’s Performance



TEST RESULTS

MODEL	VALIDATION MAE	TEST MAE
Basic Dense	2.63	2.59
1D Convolutional	2.91	3.01
Simple LSTM (Dense 16)	2.6	2.5
Simple LSTM (Dense 32)	2.98	2.64
Simple LSTM (Dense 8)	2.35	2.52
Stacked LSTM with Dropout	2.40	2.57
Bidirectional LSTM	2.46	2.60
Combination of 1D_Convnet and LSTM	3.92	3.79

CONCLUSION

Various models of deep learning were investigated and evaluated in this time series forecasting research with the Jena weather dataset. Recurrent neural network (RNN) designs, especially long short-term memory (LSTM) and bidirectional LSMs, outperform more basic models such as dense neural networks and one-dimensional convolutional networks, as the results clearly show.

The temporal dependencies in the sequential data were difficult for the 1D CNN and basic dense neural network models to capture in an efficient manner. Through processes such as pooling, these models flattened or otherwise disturbed the time series data, making it impossible for them to utilize the sequential patterns that are essential for precise anticipating.

By design, on the other hand, the recurrent connections and gating mechanisms of the LSTM and bidirectional LSTM models might preserve the temporal context and represent long-range interdependence. Bidirectional processing and recurrent dropout are two further techniques that improved the LSTM's performance.

With the lowest mean absolute error of 2.46 on the validation set, the bidirectional LSTM performed best out of all the models that were assessed. Because it could process the input both forward and backward, its bidirectionality helped it learn patterns more effectively.

For this time series challenge, the GRU model showed good computational efficiency; however, optimizing the LSTM structures through changes to hyperparameters such as unit count and dropout rates yielded better forecasting accuracy.

To summarize, the analysis emphasizes how crucial it is to use sophisticated RNN features like LSTMs and bidirectional processing to capture long-range temporal relationships. This is a prerequisite for accurate time series forecasting of sequential data, like weather data. Using the Jena weather dataset, the bidirectional LSTM model was found to be the best architecture for this forecasting task.