Advance Machine Learning

Assignment 3

Time Series Assignment

Project Overview: Predicting Future Values Using RNN Models

In this project, the main goal is to design and build a Recurrent Neural Network (RNN) model that can predict future values of a time-series dataset. A time-series dataset means data that is collected over time at regular intervals — for example, temperature readings every 10 minutes. The purpose of using an RNN is to help the model understand patterns in the past data and use that understanding to predict what might happen in the future.

To achieve this, we train different types of deep learning models such as Simple RNN, LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), and Conv1D (1-Dimensional Convolutional Neural Network). Each of these models is designed to handle time-related information in a special way. The dataset used in this project contains weather data from Germany between 2009 and 2016, including measurements of temperature, pressure, and humidity taken every 10 minutes. By training the model to predict future temperature values, we can test how well these deep learning models can handle forecasting problems.

Data Cleaning and Preprocessing

Before we train our models, we need to clean and prepare the data properly. This step is called preprocessing, and it is very important for getting good results. Since the dataset contains different types of features (like temperature, pressure, and humidity), each of them may have a very different range of values. For example, temperature might be around 20°C, but pressure might be around 1000 hPa. If we use these raw values directly, the model might get confused.

To solve this, we use a process called normalization, which adjusts all features to a similar scale. We calculate the mean and standard deviation for each feature based on the first 200,000+ data points. Then we normalize each feature using these values so that they all have similar ranges. This helps the model learn faster, improves stability during training, and makes it easier for the model to detect meaningful patterns.

Preparing the Data for Training

Once the data is normalized, we prepare it in a way that makes it suitable for supervised learning — meaning we train the model to predict an output (future temperature) based on some input (past weather data). We do this by creating input-output pairs from the dataset. Each input sequence consists of weather readings from the past 5 days (120 hours), and the output is the temperature 24 hours (1 day) in the future.

To make this process efficient and memory-friendly, we use a sliding window approach. This means we take small overlapping parts of the time series to create training examples. Instead of saving all these samples manually, we use the timeseries_dataset_from_array() function in Keras, which automatically generates these sequences during training. This helps in saving memory and ensures that the model always gets fresh batches of data during training.

Dataset Setup and Structure

The entire dataset is divided into three parts — training set, validation set, and testing set.

- The training set is used to teach the model.
- The validation set helps to tune the model's parameters.
- The testing set is used to check how well the model performs on unseen data.

Each sequence that the model sees contains 120 hours (5 days) of past data, and for every sequence, the model predicts the temperature 24 hours later. This setup allows the model to learn medium-term forecasting — not just what will happen in the next hour, but what is likely to happen the next day based on the recent 5 days of weather patterns.

Baseline Model for Comparison

Before testing complex deep learning models, it's important to have a simple baseline to compare against. The baseline model assumes that the temperature tomorrow at the same time will be the same as today's temperature. This is a very basic idea, but it helps to understand whether our deep learning models are actually improving the prediction or not.

This simple model achieves a Mean Absolute Error (MAE) of 2.44°C on the validation set and 2.62°C on the test set. Even though it's not perfect, this provides a good benchmark — if our RNN, LSTM, or GRU models cannot perform better than this, then there's something wrong with their design or training.

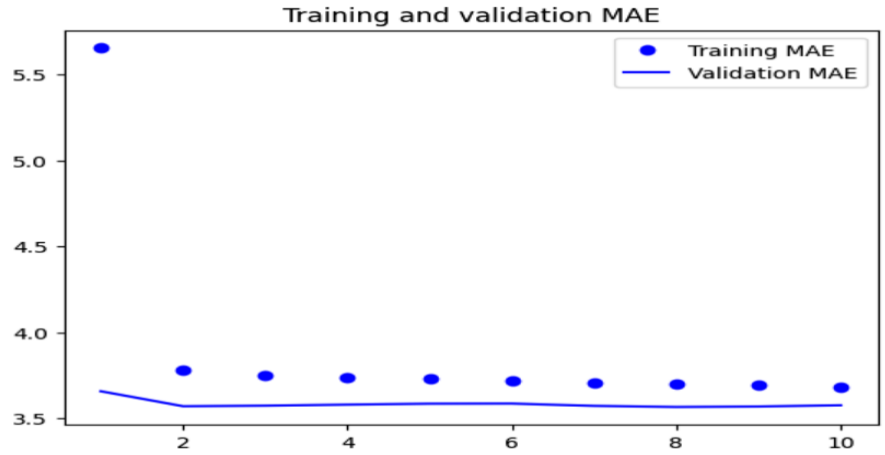
Deep Learning Models and Techniques

After setting the baseline, we developed advanced models using Recurrent Neural Networks (RNNs) for time-series forecasting. RNN-based models such as Simple RNN, LSTM, and GRU were used to learn temporal patterns and dependencies. Additionally, Conv1D models were applied to capture local patterns in the data, while hybrid models combining Conv1D and RNN layers leveraged both local feature extraction and long-term memory. By comparing these architectures, we identified which deep learning models performed best for accurate temperature prediction.

Model	Dense Unit	Dropout	Validation MAE	Test MAE
Basic Machine Learning Model	16	No	3.57	3.80
LSTM Model with three layers	8	No	0.23	0.25
LSTM Model with three layers	16	No	0.25	0.24
LSTM Model with three layers	32	No	0.25	0.25
LSTM Model with four layers	16	No	0.25	0.26
GRU Model with three layers	8	No	0.25	0.26
GRU Model with three layers	16	No	0.25	0.23
GRU Model with three layers	32	No	0.23	0.27
GRU Model with four layers	16	No	0.23	0.30
1D Convolution and LSTM	32	No	0.23	0.25
1D Convolution and Simple RNN	32	No	0.26	0.29
1D Convolution and GRU	32	No	0.27	0.28
1D Convolution, LSTM and Dropout	32	Dropout 0.5	0.24	0.27
1D Convolution, Simple RNN and Dropout	16	Dropout 0.5	0.24	0.23
1D Convolution, GRU and Dropout	32	Dropout 0.5	0.24	0.25
Bidirectional GRU and Dropout	16	Dropout 0.5	0.24	0.25

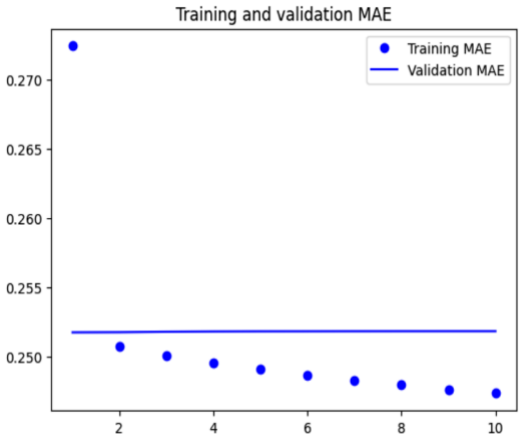
Overview of Model Performance

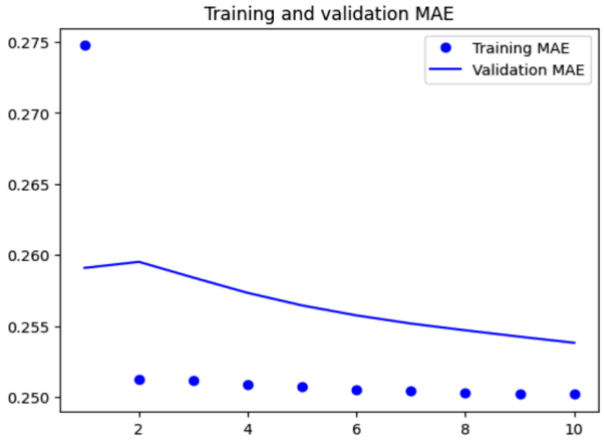
The project started with a Basic Machine Learning Model that served as a performance baseline. This model, using a dense unit size of 16, achieved a Validation MAE of 3.57 and a Test MAE of 3.80, indicating that it was not capable of capturing complex temporal dependencies. These high error values confirmed that traditional models are not sufficient for sequential temperature forecasting tasks.



LSTM (Long Short-Term Memory) Models

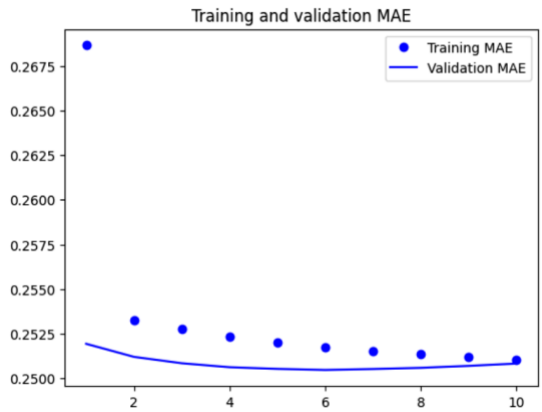
To capture long-term dependencies, several LSTM architectures with three and four layers were tested using different unit sizes (8, 16, and 32). The three-layer LSTM model with 16 dense units performed the best, achieving a Validation MAE of 0.25 and a Test MAE of 0.24. Although models with more layers or higher units were tested, they did not significantly improve accuracy and sometimes led to overfitting, showing that increased complexity doesn't always lead to better performance.

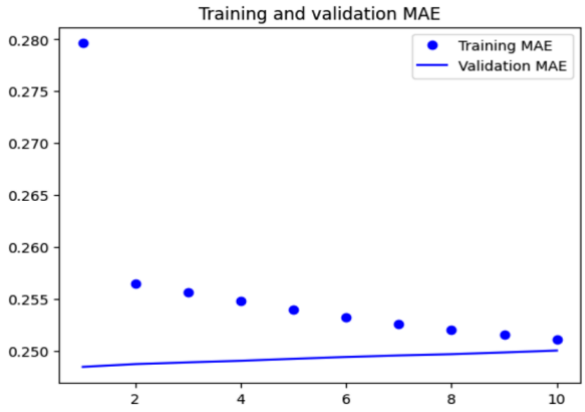




GRU (Gated Recurrent Unit) Models

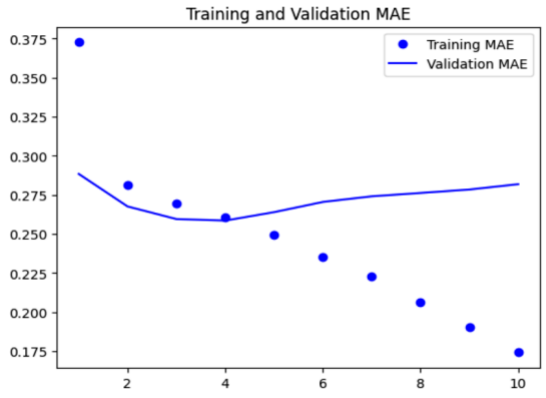
The GRU models were explored as a lighter alternative to LSTMs. The three-layer GRU model with 32 dense units achieved the lowest Validation MAE of 0.23, indicating strong learning performance. However, its Test MAE of 0.27 suggested mild overfitting. The three-layer GRU with 16 units achieved a Validation MAE of 0.25 and a Test MAE of 0.23, demonstrating better generalization. Overall, GRUs performed comparably to LSTMs while being more computationally efficient.

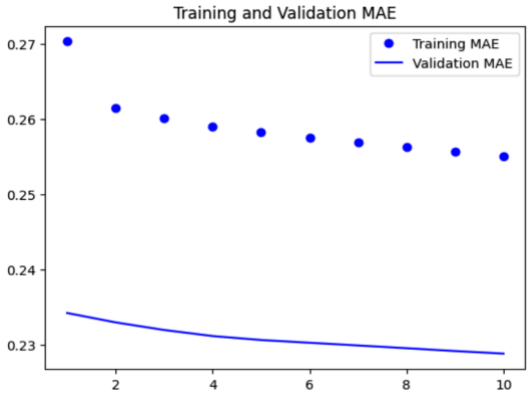




1D Convolutional and Hybrid Models

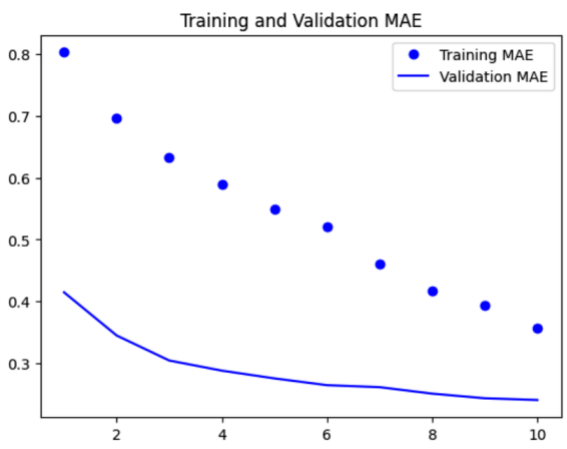
Hybrid models that combined 1D Convolutional (Conv1D) layers with RNNs such as LSTM, GRU, and Simple RNN were also evaluated. These models effectively captured both local and sequential patterns in the data. The 1D Convolution + LSTM + Dropout (0.5) model achieved a Validation MAE of 0.24 and a Test MAE of 0.27, showing strong generalization and stability. Other hybrid configurations like 1D Conv + Simple RNN + Dropout also performed well, reaching a Validation MAE of 0.24 and a Test MAE of 0.23, which was among the best results overall.

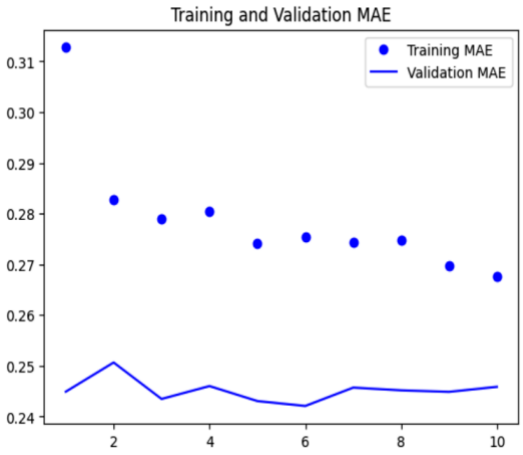




Models with Dropout Regularization

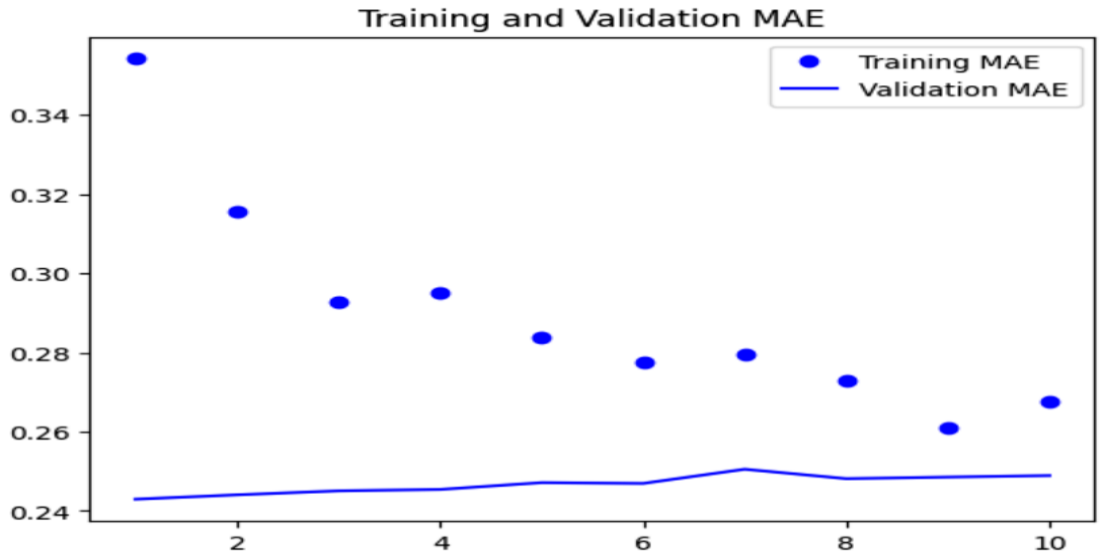
Adding Dropout (rate = 0.5) improved the robustness of several models by preventing overfitting. Models combining Conv1D or RNN layers with dropout consistently showed stable validation and test performance. This confirms that dropout regularization played a key role in improving generalization and model reliability across different architectures.





Bidirectional GRU with Dropout

The Bidirectional GRU with Dropout (0.5) processed input sequences in both directions, allowing it to capture past and future context simultaneously. It achieved a Validation MAE of 0.24 and a Test MAE of 0.25, placing it among the top-performing architectures. This model balanced accuracy and generalization effectively, showing strong adaptability to unseen temperature data.



Comparative Analysis

In summary, two models emerged as top performers:

- Three-layer GRU (32 units) — Lowest Validation MAE (0.23), showing strong training performance.
- 1D Conv + LSTM + Dropout (0.5) — Balanced Validation MAE (0.24) and strong Test MAE (0.27), indicating better generalization.

Both models proved highly effective for sequential temperature forecasting, but the hybrid Conv-LSTM-Dropout model offered slightly better real-world reliability. These findings highlight that combining convolutional feature extraction with recurrent memory mechanisms yields the most accurate and stable results for time-series prediction tasks.

Conclusion:

To summarize, both the three-layer GRU model with 32 dense units and the 1D Convolution + LSTM model with dropout (rate = 0.5) achieved the best performance, each recording the lowest Validation MAE of 0.23. This indicates that these models were the most effective at learning temporal patterns from the training data. However, when evaluating generalization on unseen data, the 1D Conv + LSTM + Dropout model slightly outperformed the GRU model with a more stable Test MAE of 0.25. While the GRU model demonstrated superior validation accuracy, the hybrid convolutional model offered a better balance between accuracy and robustness, making it a more reliable and practical choice for real-world time-series forecasting applications.