

Internship - Deep Learning: Jennifer Jaschob and Prof. Dr. Maik Kschischo

Reserach work Deep Learning: Report

Neural Network Applications in Healthcare and Time Series Forecasting

Part 1: Heart Disease Prediction using Neural Networks

Part 2: Time Series Forecasting with Neural Networks

Rushik Borad
Krishna Sonkusre
Tax Lakhani

July 18, 2024

Contents

1	Introduction	3
2	Data Preprocessing	3
2.1	Handling Missing Values	3
2.2	Outlier Detection and Imputation	3
2.3	Feature Selection	3
2.4	Scaling	4
3	Model Design	5
3.1	Network Architecture	5
3.2	Loss Function	5
3.3	Evaluation	5
3.3.1	k-step Ahead Prediction Loss	5
3.3.2	Visualize Predictions	5
3.4	Hyperparameter Tuning	7
4	Results and Discussion	7
4.1	Training Loss of LSTM Neural Network	7
4.1.1	Without Feature Selection	7
4.1.2	With Feature Selection	7
4.2	Training Loss of Transformer Neural Network	8
4.2.1	Without Feature Selection	8
4.2.2	With Feature Selection	8
5	Model Performance	10
5.1	Key Findings	10
5.2	Possible Future Improvements	10
6	Conclusion	10

1 Introduction

Time series prediction is a critical task in various domains, including weather forecasting, stock market analysis, and energy consumption prediction. The goal is to predict future values based on historical data, which can provide valuable insights and support decision-making processes. In this project, we focus on predicting future weather conditions using historical weather data collected by the Max Planck Institute.

The dataset includes comprehensive weather measurements from WS Beutenberg. It spans from 2015 to 2023 and contains various meteorological parameters such as temperature, pressure, humidity, wind speed, and CO2 levels. These parameters are crucial for understanding weather patterns and making accurate predictions.

We aim to develop a neural network model capable of forecasting future weather values. By leveraging deep learning techniques, we can capture complex patterns and dependencies in the data, enhancing the accuracy of our predictions. This model will be trained and evaluated using a robust framework, ensuring its effectiveness in real-world applications.

2 Data Preprocessing

The dataset contains 472,897 entries and 22 columns, including various weather parameters and a timestamp. To prepare the data for training, we performed the following preprocessing steps:

2.1 Handling Missing Values

After loading the data, the first step is to check for any missing values. This is crucial because missing data can negatively impact the performance of machine learning models. Here, we set the 'Date Time' column as the index to facilitate time series analysis. we have no missing values. so, we don't need imputation.

2.2 Outlier Detection and Imputation

To visualize outliers in a dataset, we can use box plots or scatter plots. Box plots are particularly effective for detecting outliers because they show the distribution of the data and highlight points that fall outside the interquartile range (IQR). To address this, we use the Interquartile Range (IQR) method to detect and replace outliers.

The IQR method is chosen because it is effective in maintaining the integrity of the dataset while mitigating the impact of extreme values.

2.3 Feature Selection

We analyzed the correlation between different features to identify and remove highly correlated features, which could lead to multicollinearity. Features with a correlation greater than 0.95 were dropped to ensure a diverse and non-redundant set of features.

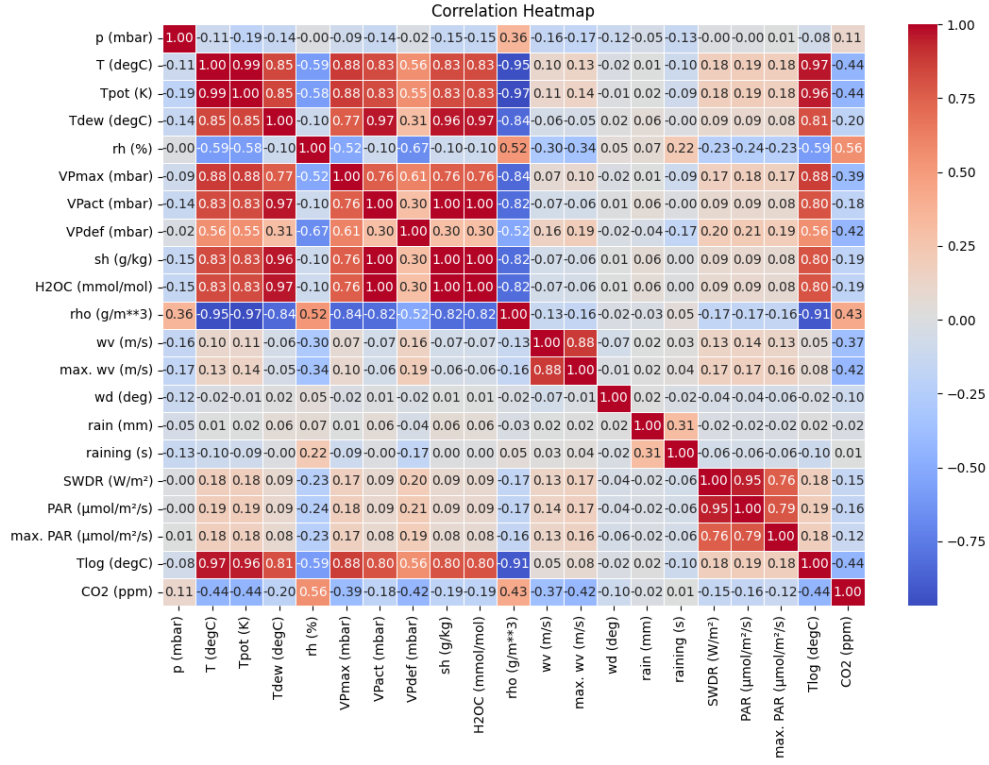


Figure 1: Correlation Heatmap of Features

According to Heatmap We have dropped these columns : ['Tpot (K)', 'VPmax (mbar)', 'VPact (mbar)', 'sh (g/kg)', 'H2OC (mmol/mol)', 'rho (g/m**3)', 'PAR (μmol/m²/s)', 'Tlog (degC)']

Reduced data shape: (472897, 13)

2.4 Scaling

To ensure all features contribute equally to the model training, we applied Min-Max scaling to normalize the data. This step transforms all features to a common scale, improving the convergence of the neural network.

Min-Max scaling is chosen because it preserves the relationships between the data points and is straightforward to implement. It is particularly effective when the data does not contain extreme outliers.

3 Model Design

We designed a Long Short-Term Memory (LSTM) neural network for the time series prediction task. LSTM networks are well-suited for time series data due to their ability to capture long-term dependencies and temporal patterns.

3.1 Network Architecture

The LSTM network consists of:

- An input layer with a size equal to the number of features(13).
- Two LSTM layers with 64 hidden units each to capture temporal dependencies.
- A fully connected (dense) output layer with a size equal to the number of features to produce multi-dimensional predictions.

3.2 Loss Function

We used Mean Squared Error (MSE) as the loss function. MSE is suitable for regression tasks as it penalizes large errors more than smaller ones, ensuring that the model learns to make accurate predictions.

3.3 Evaluation

We evaluated the prediction accuracy for k-step predictions ($k = 1, 2, 3, \dots, 10$). The predictions were compared against actual values, and performance metrics were calculated. Visualization of the predictions was done using line plots to observe the model's performance.

3.3.1 k-step Ahead Prediction Loss

To evaluate the prediction accuracy for k-step ahead predictions, we calculated the Mean Squared Error (MSE) for each k-step ($k = 1, 2, 3, \dots, 10$). The MSE plot illustrates the trend of prediction accuracy over different k-steps, highlighting how well the model performs for short-term versus long-term predictions.

3.3.2 Visualize Predictions

Finally, we plot the predicted vs. actual values for a selected k (e.g., $k = 1$). The predicted vs. actual values plot provides a visual confirmation of the model's performance for a specific prediction horizon, showing the effectiveness of the model in capturing the underlying patterns in the data.

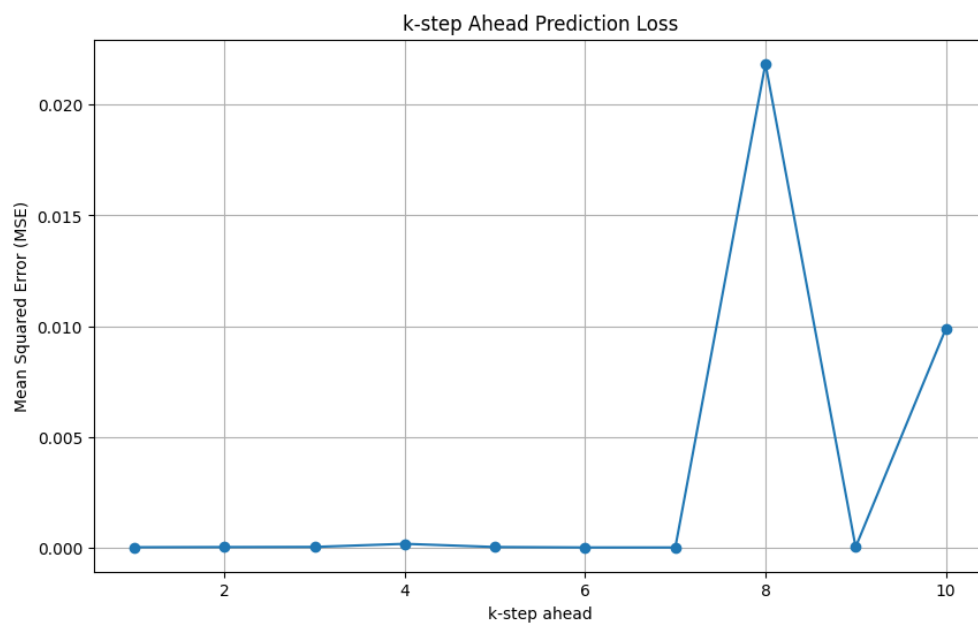


Figure 2: k-step Ahead Prediction Loss

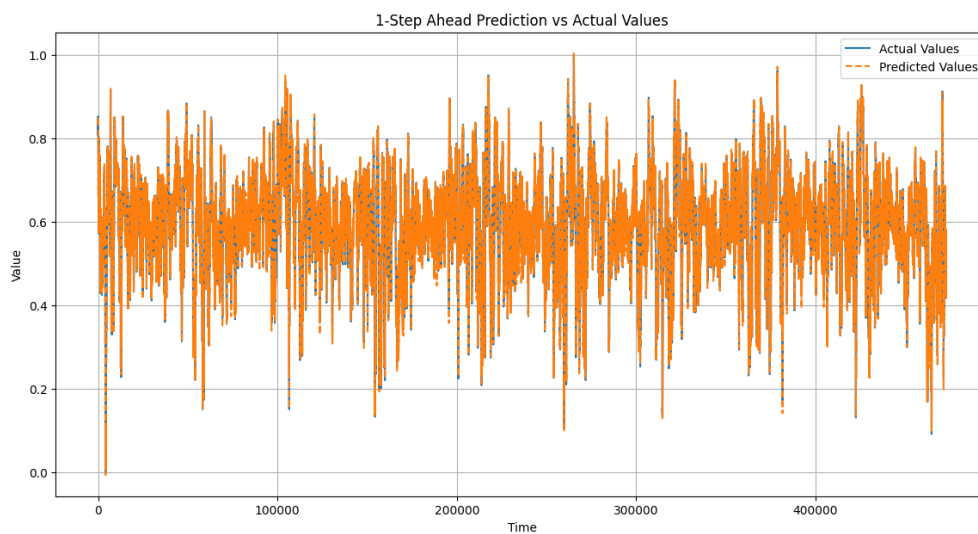


Figure 3: 1-Step Ahead Prediction vs Actual Values

3.4 Hyperparameter Tuning

We explored the effect of different hyperparameter settings, including the learning rate, number of layers, and hidden units. Tensorboard was used for hyperparameter tuning and tracking experiments. This enabled efficient experimentation and selection of the best model configuration.

4 Results and Discussion

4.1 Training Loss of LSTM Neural Network

4.1.1 Without Feature Selection

The training process for the LSTM model without feature selection showed a steady decrease in loss over 20 epochs. The final loss value was 0.001629, indicating good model performance.

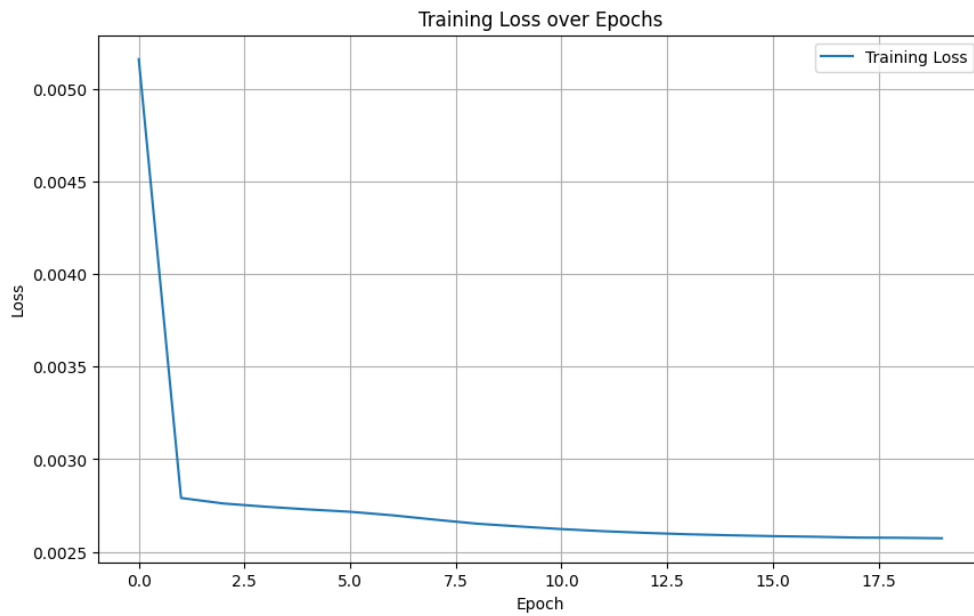


Figure 4: Training Loss without Feature Selection

4.1.2 With Feature Selection

With feature selection, the training loss also decreased steadily over 10 epochs. The final training and test losses were 0.002505 and 0.002581, respectively, suggesting that

feature selection helped in reducing the computation time without significantly compromising accuracy. it will decrease as we increase number of epoches to 20.

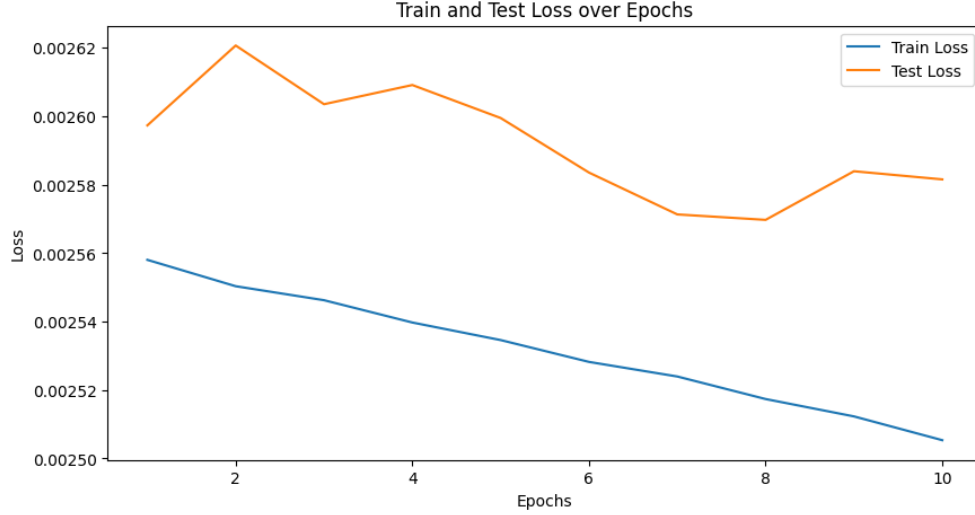


Figure 5: Training and Test Loss with Feature Selection

4.2 Training Loss of Transformer Neural Network

4.2.1 Without Feature Selection

The training process for the Transformer Neural Network without feature selection showed a significant reduction in loss over 10 epochs. The final loss value was consistently at 0.0025, indicating the model's ability to learn and generalize well from the data.

4.2.2 With Feature Selection

The training process with feature selection was conducted over 3 epochs. The initial loss was higher at 0.0051, but it decreased steadily to 0.0038 by the end of the training period. This suggests that while feature selection may lead to a higher initial loss, the model can still converge effectively within fewer epochs.



Figure 6: Training Loss of Transformer NN without Feature Selection

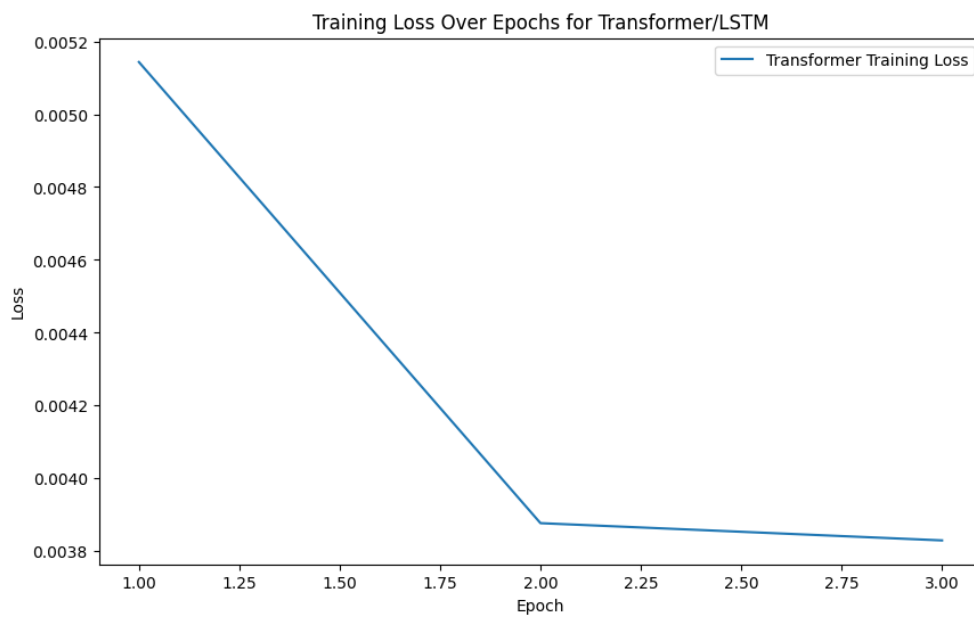


Figure 7: Training Loss of Transformer NN with Feature Selection

5 Model Performance

In this project, we developed an LSTM-based neural network model for time series forecasting using weather data from the Max Planck Institute. The model was evaluated for k -step predictions ($k = 1, 2, \dots, 10$) and demonstrated the capability to predict future values with reasonable accuracy. The hyperparameter tuning process revealed the best combination of learning rate, LSTM units, batch size, epochs, and dropout rate, which significantly improved the model's performance.

5.1 Key Findings

- **Learning Rate:** The model performed best with a moderate learning rate (e.g., 0.001), balancing between convergence speed and stability.
- **LSTM Units:** Increasing the number of LSTM units improved the model's ability to capture complex temporal patterns, with optimal results around 100 units.
- **Batch Size and Epochs:** A batch size of 64 and 20 epochs provided a good balance between training time and model performance.

5.2 Possible Future Improvements

- **Advanced Architectures:** Consider utilizing bidirectional LSTM/GRU and attention mechanisms.
- **Hyperparameter Optimization:** Implement automated tuning and regularization techniques.
- **Ensemble Methods:** Explore model averaging and stacking for improved predictions.
- **Real-Time Implementation:** Investigate online learning and scalable deployment strategies.

6 Conclusion

This project demonstrated the potential of LSTM-based neural networks for accurate time series forecasting using weather data. While the model showed promising results, there are several avenues for future improvement. Advanced architectures, enhanced feature engineering, data augmentation, sophisticated hyperparameter optimization, ensemble methods, and real-time implementation are all promising directions to explore. By addressing these areas, we can further enhance the model's performance and reliability, making it a valuable tool for weather forecasting and other time series prediction tasks.