Predicting Weather using Deep Learning

Emmanuel Marcial, Rohin Sampeur, Michael Zermeno
ECE 4715
Cal Poly Pomona

Abstract

Accurate weather prediction plays a critical role in areas such as agriculture, transportation, and disaster management. This project will focus on exploring the applications of deep learning for weather prediction based on two weather datasets, Jena Climate dataset and the Seattle Weather dataset. The system in this project uses a combination of data augmentation and Deep Learning models such as a Feed-Forward Neural Network (FFNN), and a Recurrent Neural Network (RNN). The FFNN is used for weather category classification in the Seattle dataset, its accuracy performance is benchmarked against traditional machine learning models such as Random Forest and XGBoost. The RNN is used for a regression task, predicting temperature in both the Seattle and Jena dataset. An important aspect of deploying the RNN model involves the restructuring of the given datasets into a sequential format. This approach allows RNNs to analyze and use recurring patterns more effectively.

Overall this project explores the applications of deep learning in capturing patterns in weather data, over traditional machine learning approaches. The final results show that the RNN model excels in forecasting temperature, with low MAE on both datasets, and the FFNN model is able to employ a high accuracy in classification tasks. The findings of this project also highlight the impact of the quality of a dataset, such as the size or feature correlations.

1. Introduction

Traditional machine learning models and other approaches have long been used for weather prediction. Exploring the applications of deep learning could allow new opportunities to emerge that will model weather patterns more effectively than traditional models. In this project we investigate the application of two popular Deep Learning models, Recurrent Neural Networks and Feed-Forward Neural Network. As mentioned earlier these models were deployed on two datasets, Jena Climate and Seattle Weather. The decision to use RNN was made due to the popularity of RNN being used for time series forecasting tasks, which weather prediction falls under. Because of this our RNN model was tasked with regression, prediction of the temperature for the Jena Climate dataset, and the temperature range, which is a feature we created, for the Seattle Weather dataset. The FFNN model was applied to classify weather conditions for our Seattle datasets with category labels, whether it's "rainy", "fog", "sunny" etc. The accuracy of that FFNN model is then compared to the accuracy of 3 popular machine learning models that are often used for classification tasks, Gradient Boosting (GB), XGBoosting (XGB), and Random Forest (RF).

For the FFNN and Machine Learning models we use evaluation metrics such as accuracy, loss and a classification report, which provides more metrics such as f1-score, recall and precision. The RNN model is evaluated using different metrics, due to it being used for regression tasks instead of classification. These metrics are Mean Absolute Error (MAE) and our loss Mean Squared Error (MSE), we also create plots comparing Actual vs Predicted values to analyze the accuracy of the RNN model's predictions.

Before the models were deployed on the dataset and fine-tuned, work on analyzing the data in both datasets was done first. This was done to verify the correlations between the features, find potential skewed data which may impact the performance of our models, and find out if there are other features we could potentially create. Analyzing these datasets demonstrated how important data augmentation is when it comes to predictive modeling, because characteristics such as skewed or imbalanced data and feature engineering can greatly impact the performance of a model.

2. Related Work

In the development of this weather prediction system, research papers provided valuable insights into machine learning techniques. These studies helped give us a foundation for our project which led to various key factors of the project's design. These papers were also the main inspiration for using machine learning models to benchmark our deep learning models against.

Weather Prediction Performance Evaluation on Selected

Machine Learning Algorithms by Muyideen AbdulRaheem explored the use of a web application built using the Spyder IDE using the Pandas and Scikit-learn libraries for data manipulation and machine learning. This system implemented three different classification algorithms to predict rainfall. The Decision Tree model achieved 100% accuracy for the dataset, the K-Nearest Neighbor (KNN) achieved 78% accuracy and the Logistic Regression achieved 93% accuracy. The data set used for training and testing consisted of 39 years of forecast data from Seattle-Tacoma International Airport. The data captures rainfall patterns and includes the features of date, precipitation, maximum temperature, minimum temperature, and rain. During preprocessing, normalization techniques were applied in order to remove or modify duplicate entries. This was done to ensure data consistency. The data was then split into training and testing sets to evaluate model performance. The data modeling phase for this project involved the various machine learning algorithms, listed earlier, to train the model on the processed dataset. The trained models were then evaluated on the testing set to test the performance and ensure that predictions were accurate.

Rainfall Prediction using Machine Learning & Deep Learning Techniques by Cmak Zeelan Basha used an artificial neural network to predict rainfall in India. Their neural network contained an Autoencoder and a Multilayer Perceptron(MLP). The auto encoder had 3 layers: the input, the output and a hidden layer. It was used to extract non-linear features from the data set and then send them to the MLP. The MLP was also connected to the auto encoder using a sigmoid function. Overall, they were able to get a working model that predicted future data using past data. The data was split into training and testing data, in order for the mean error to be calculated at each iteration. At each iteration the best epoch value was also checked. At the very last stage of their system, the training and testing results were then combined in order to find the best results.

The project in *Weather Forecasting Using Machine Learning Algorithms* by Nitin Singh utilized a Raspberry Pi along with low-cost sensors to collect weather data, making the system affordable and accessible. The machine learning algorithm that they implemented was a Random Forest Classification, which involves the combination of multiple decision trees into a single model to improve prediction accuracy. Their dataset was split into 75% for training and 25% for testing in order to evaluate the model's performance. The dataset used in this project uses 20 years of weather data from Delhi, India and included parameters such as temperature, humidity, pressure, and rainfall. Additionally they used the following 3 python modules: sense.py, frontend.py and backend.py. The software runs on the Raspbian OS, and development was carried out using Spyder 3, an Integrated Development Environment (IDE) which is designed for machine learning applications. Overall, the model achieved an accuracy of 87.9%. Further analysis revealed that humidity was the most significant variable in making accurate predictions.

The work from Suri babu Nuthalapati, *Accurate Weather Forecasting with Dominant Gradient Boosting using Machine Learning*, was especially interesting due to the unique aspect of using gradient boosting in their machine learning models. This is basically when multiple models are combined in order to create a more accurate model. For the dataset used, the model had to predict the weather outcomes in Seattle, similar to what we are hoping to achieve. The different possible outcomes were Drizzle, Rain, Sun, Snow and Fog. Some other features included Precipitation, Max and Min Temperatures and then wind speed. The author also used Seaborn, a python library, to create correlations of all of the variables which allowed them to analyze weather trends and see how each variable affected one another. This paper was the main inspiration for using popular machine learning models as benchmarks against our Feed Forward Neural Network model.

3. Proposed Approach

3.1. Architecture Details

The approach of this project uses a Recurrent Neural Network (RNN) for time-series regression tasks and a Feed-Forward Neural Network (FFNN) for structured data classification tasks. The summary of the architecture of the RNN model is shown in Figure 1, while the summary of the architecture of the FFNN model is illustrated in Figure 2.

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 7, 32)	1,248
dropout (Dropout)	(None, 7, 32)	0
simple_rnn_1 (SimpleRNN)	(None, 16)	784
dropout_1 (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17

Figure 1. Summary of the architecture of the RNN model used for regression tasks to predict temperature and temperature range.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	320
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 5)	165

Figure 2. Summary of the architecture of the FFNN model used for weather type classification.

The RNN model consists of two SimpleRNN layers,

with 32 and 16 units, respectively, both are using the tanh activation function, there are also two Dropout layers used to reduce over fitting and allow the model to generalize better, and finally a Dense output layer. The input to the RNN model consists of sequences of 7 consecutive days (time steps) for both the Jena dataset and the Seattle dataset. It processes features to predict temperature range with the Seattle dataset, and temperature with the Jena dataset. The RNN model uses the Mean Squared Error (MSE) loss function because of its assigned regression task.

The FFNN model consists of two hidden layers, with 64 and 32 units, respectively, both using the relu activation function, there are two Dropout layers once again used to reduce overfitting, and finally an output layer using the softmax activation function due to this model being used for a multiclass classification task. The model processes weather features from the Seattle dataset to predict the category of the weather type. The FFNN model uses Categorical Crossentropy loss because of its assigned classification task.

Before we applied the models on our datasets, we first analyzed the datasets to evaluate their quality and effectiveness of the features. This was done to encounter potential challenges that the models can have with regression or classification tasks with the datasets. For example in Figure 5 it can be seen that there is a high frequency of rain and sun weather categories in the dataset, this is a case of an imbalanced dataset. This was determined to be fine though due to the realistic expectations of weather types, in the real world rain and sun are more present in most locations. In Figure 3 we can see a correlation matrix for the Jena dataset, it can be seen that there a strong correlations between other features and our target variable , temperature (degC), compared to the weaker correlations in the Seattle dataset it highlights a potential challenge for the RNN model predicting temperature range with the Seattle dataset.

3.2. Baseline Architecture

The Baseline architectures of our project also includes traditional machine learning models such as Random Forest and XGBoost. Their detailed classification report metrics are presented in Figures 6 and 7, note that these machine learning models struggle with predicting minority classes in the Seattle dataset, the impact of this on our deep learning models will also be explored later. The starting points of our RNN model involved a single SimpleRNN layer with less units and Dropout layers, similarly with our FFNN model there was originally a single hidden layer with less units and Dropout layers. From there fine tuning was done by adding more units and layers until we came to the final version of the models that we determined to have the best performance.

Looking at the metrics for our traditional machine learning models such as Figure 7, it can be seen that there is a

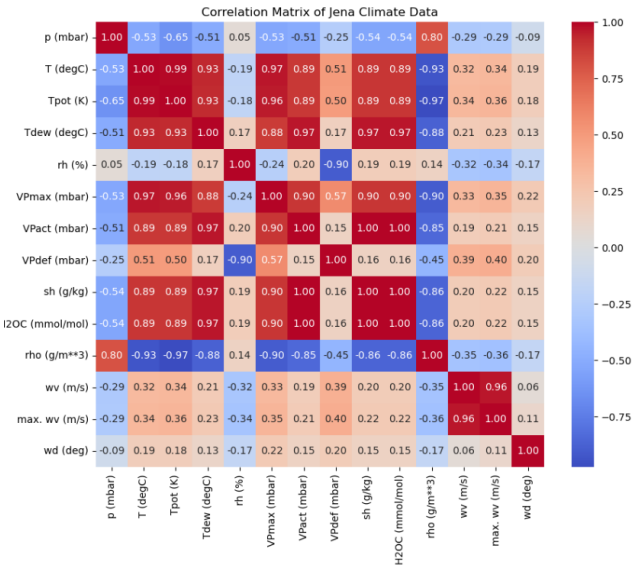

Figure 3. Correlation matrix for features in the Jena dataset.

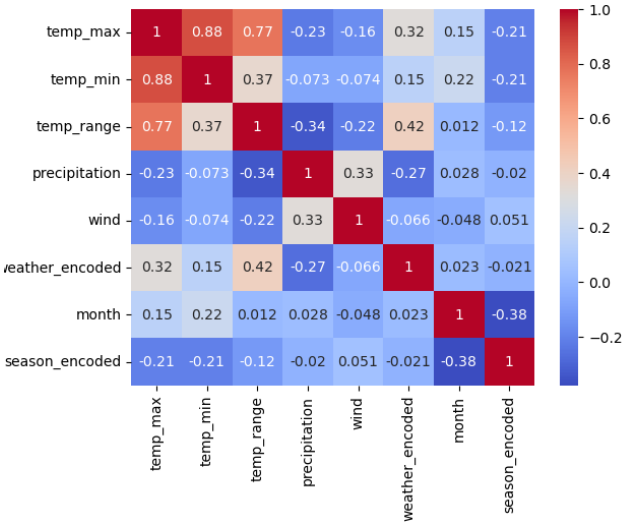

Figure 4. Correlation matrix for features in the Seattle dataset.

struggle to predict certain classes which are minority categories, as mentioned before this is likely due to the Seattle dataset being imbalanced, but again this reflects real world scenarios.

Figure 8 is another example of data analyzing that was done on the Seattle dataset. From this plot it can be seen that there were seasonal trends involved. It indicated higher precipitation in winter months and lower otherwise, this particular analysis encouraged us to create features such as season and month and explore those features correlations with the rest of the dataset and its impact on our deep learning models.

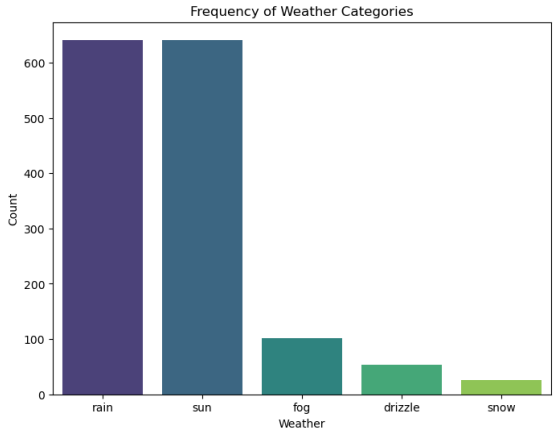

Figure 5. Bar plot showing the frequency of different weather categories in the Seattle dataset.

Random Forest Accuracy: 81.57%

Random Forest Classification Report:

	precision	recall	f1-score	support
drizzle	0.20	0.11	0.14	9
fog	0.40	0.08	0.13	25
rain	0.93	0.92	0.92	120
snow	0.40	0.25	0.31	8
sun	0.78	0.95	0.85	131
accuracy			0.82	293
macro avg	0.54	0.46	0.47	293
weighted avg	0.78	0.82	0.78	293

Figure 6. Evaluation metrics for Random Forest on the Seattle dataset.

XG Boost Accuracy: 80.89%

XG Boost Classification Report:

	precision	recall	f1-score	support
drizzle	0.17	0.11	0.13	9
fog	0.50	0.20	0.29	25
rain	0.92	0.92	0.92	120
snow	0.33	0.25	0.29	8
sun	0.79	0.91	0.84	131
accuracy			0.81	293
macro avg	0.54	0.48	0.49	293
weighted avg	0.78	0.81	0.79	293

Figure 7. Evaluation metrics for XGBoost on the Seattle dataset.

4. Results / Experiments

4.1. Data

The Jena Climate dataset is a very large dataset that includes over 450,000 entries , for the sake of training time it

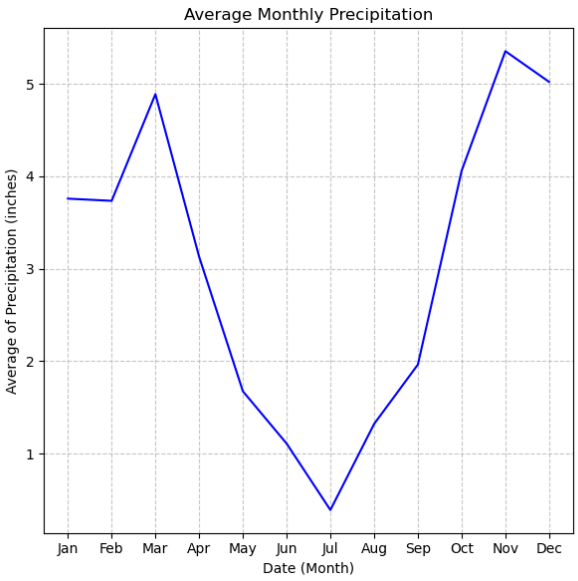

Figure 8. Average monthly precipitation for the Seattle dataset.

is downsampled to 10,000 before we train it with the RNN model. The Jena dataset contains weather features like temperature, pressure, and humidity. The Seattle Weather dataset contains 1,400 entries and includes features such as precipitation, temperature, and wind and engineered features such as season and month. We split 20% of the Seattle dataset for testing and the other 80 % was used for training the Feed Forward Neural Network model, the same data split was used on the Jena Climate dataset for training and testing the Recurrent Neural Network model.

4.2. Metrics

We evaluated the regression task of our RNN model using Mean Absolute Error (MAE) and Mean Squared Error (MSE). For our FFNN and Machine Learning models' classification task, we used a classification report which displays precision, recall, F1-score, and accuracy. Detailed equations and descriptions of these metrics are below.

Regression Metrics The regression metrics used in this project are defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

where n is the total number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value. The Mean Absolute Error (MAE) measures the average magnitude of the prediction errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

where n is the total number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value. The Mean Squared Error (MSE) penalizes larger errors more than smaller ones due to squaring, this does make it more sensitive to outliers.

Classification Metrics The classification metrics used in this project are defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. Accuracy is the proportion of correctly classified predictions out of the total number of predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

Precision quantifies how many of the predicted positive cases are true positives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

Recall measures the ability of the model to correctly identify all positive cases.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

The F1-score is the mean of precision and recall.

The FFNN model uses the Categorical Crossentropy loss function, defined as:

$$\text{Loss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{7}$$

where C is the number of classes, y_i is the binary indicator, and $\hat{y}_i$ is the predicted probability for i. This loss function penalizes incorrect predictions with high "confidence", which helps encourage the model to refine its predictions.

Activation Functions The activation functions used in our models are defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8}$$

The tanh function maps input values, which enables the model to capture both positive and negative feature relationships.

$$\text{ReLU}(x) = \max(0, x) \tag{9}$$

The relu function introduces the property of nonlinearity by setting negative inputs to zero.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{10}$$

Softmax normalizes the outputs into probabilities, it ensures that the sum of probabilities across all classes equals 1.

4.3. Results

4.3.1 RNN Results on Jena dataset

The RNN model achieved high accuracy in predicting temperature for the Jena dataset. Scatter plots comparing actual and predicted values are shown below. In Figure 9 we can see the points closely follow the diagonal line which indicates strong performance with the model. And in Figure 10 we can see the models Actual vs Prediction temperatures over the first 100 samples. The model's performance over epochs is also shown below, in Figure 11 we can see that the loss convergence indicates the model has effective training, there is also no sign of over fitting meaning the RNN model is generalizing well.In Figure 12 we can see our MAE values, these MAE values are consistent which likely means the performance of our RNN model is high and stable.

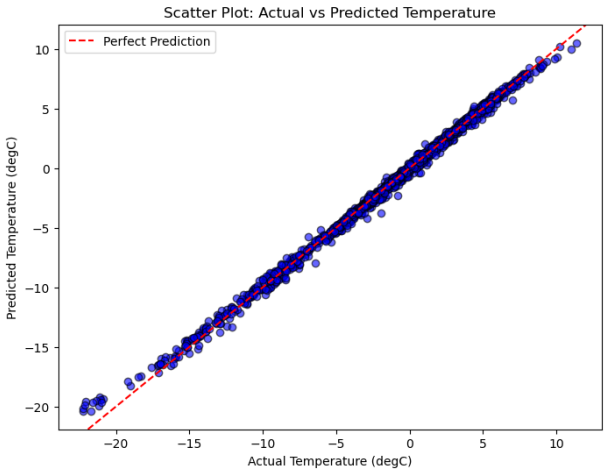

Figure 9. Scatter plot comparing actual and predicted temperatures for the Jena dataset.

4.3.2 RNN Results on Seattle dataset

The FFNN model also performed reasonably well on the Seattle dataset for predicting temperature range even though the dataset itself is of low quality as mentioned before. Figures 13 and 14 are the scatter plots for our Actual vs Predicted values for the Seattle dataset. It can be seen that the model has reasonable accuracy as it follows the diagonal line and the real values well, but there is a clear struggle of

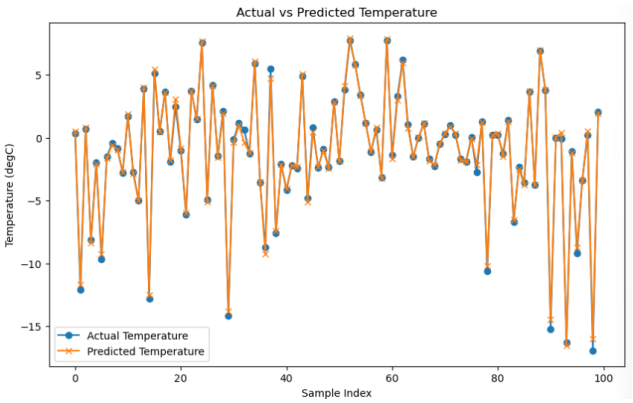

Figure 10. Line plot comparing actual and predicted temperature values over 100 samples.

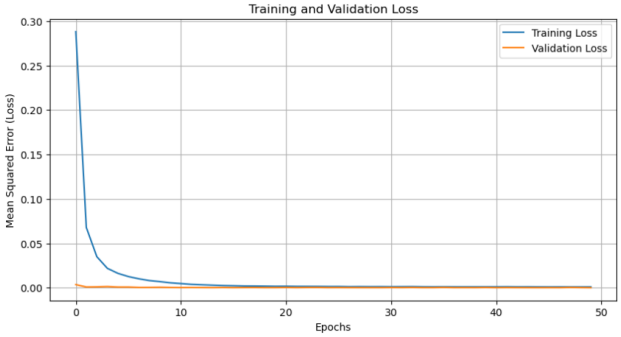

Figure 11. Training and validation loss over epochs on the Jena dataset.

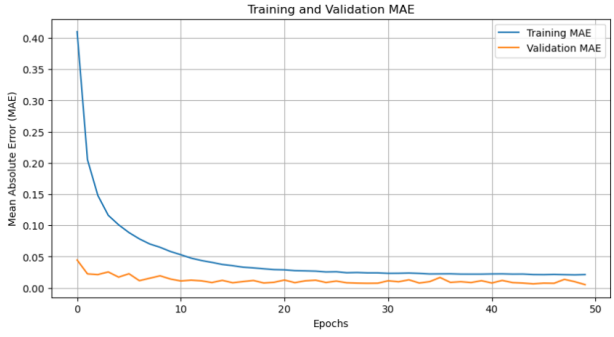

Figure 12. Training and validation MAE over epochs for the RNN model on the Jena dataset.

predicting outlier values such as high temperatures, as mentioned before this is likely due to the Seattle dataset being imbalanced and its small size, the rarity of the presence of high temperature values in the dataset is likely affecting the model's ability to predict higher values.

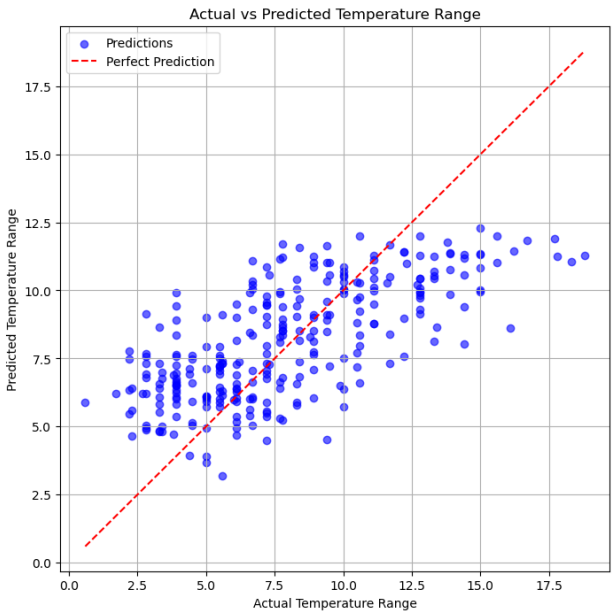

Figure 13. Scatter plot comparing actual and predicted temperature ranges for the Seattle dataset.

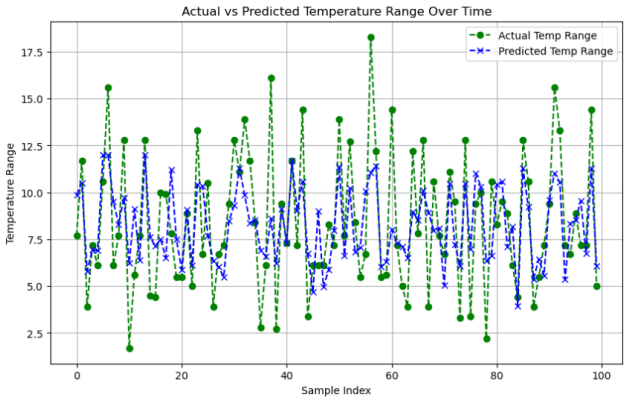

Figure 14. Line plot comparing actual and predicted temperature ranges over 100 samples.

4.3.3 FFNN Results

For the Seattle dataset, the FFNN achieved the highest classification accuracy of 88%, outperforming the benchmark Machine Learning models, as shown in Figure 15 although there is some slight fluctuation. Training and validation loss trends are also shown in Figure 16. One thing to note is that the validation set seems to outperform the training set, but this is likely due to how Keras models work, where during testing regularization mechanisms such as Dropout layers are turned off, this is reflected in our FFNN model. Confusion matrices for specific weather categories are also shown in Figures 17, 18, and 19.

We can see in the confusion matrices for our FFNN

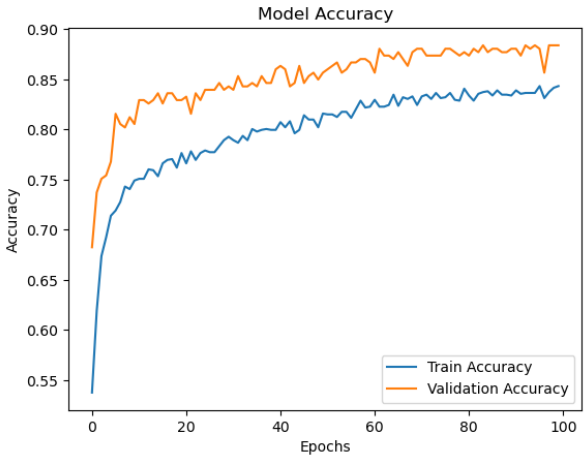

Figure 15. Training and validation accuracy over epochs for the FFNN model.

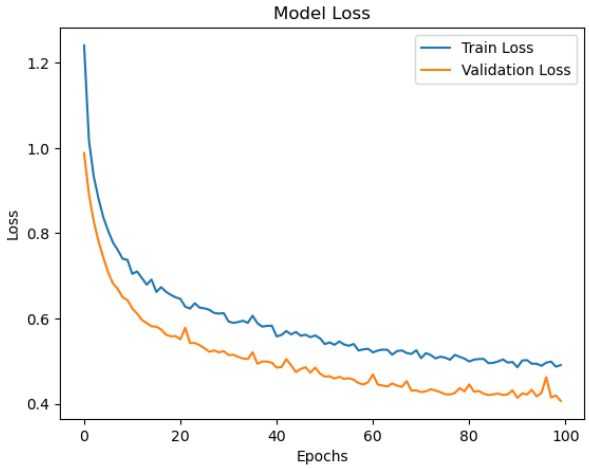

Figure 16. Training and validation loss over epochs for the FFNN model.

model that our model achieves high precision and recall for rain and sun, overall the FFNN model is pretty accurate, but it can also be noted that it struggles to predict minority classes such as fog. As noted before this is likely due to the small presence of fog in the Seattle dataset, once again data imbalance is likely having a impact here.

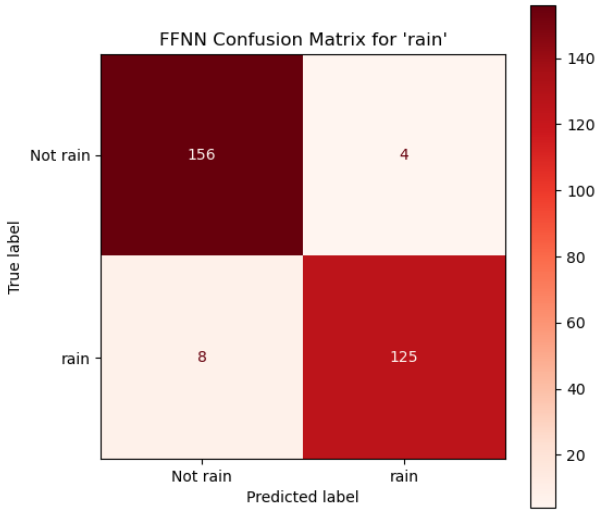

Figure 17. Confusion matrix for rain using FFNN.

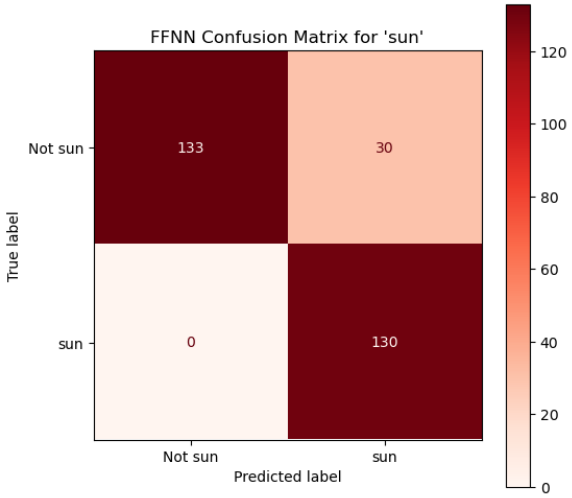

Figure 18. Confusion matrix for sun prediction using FFNN.

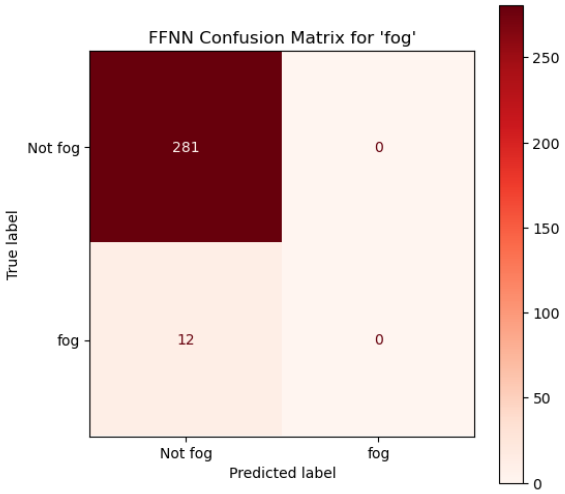

Figure 19. Confusion matrix for fog prediction using FFNN.

We can compare the confusion matrices of the RF machine learning model with our deep learning model. For example, in 20 we can see that the RF model's prediction for rain has 10 false positives and 10 false negatives, compared to the FFNN model, which has 4 false positives and 8 false negatives. Similarly in 21 we can see that the RF model has 32 false positives and 12 false negatives, where the FFNN model performs better once again. In 22 we can see that the low distribution of fog category in the dataset is also having an impact on the RF model, where it also has trouble predicting fog, the RFs classification report also displays this. But overall the FFNN model has better accuracy with predicting the weather type with our seattle dataset against all of the benchmark machine learning models we used.

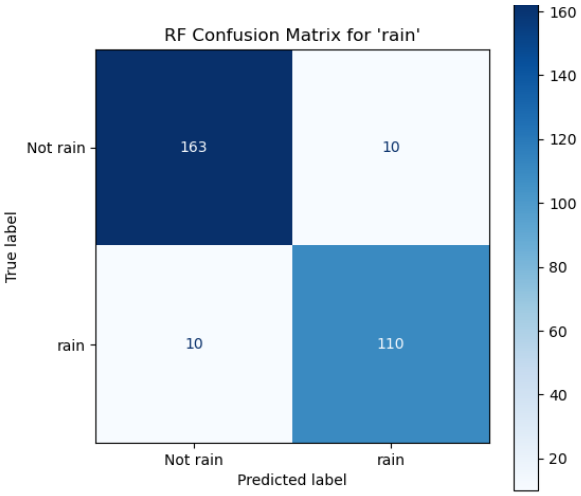

Figure 20. Confusion matrix for rain prediction using RF.

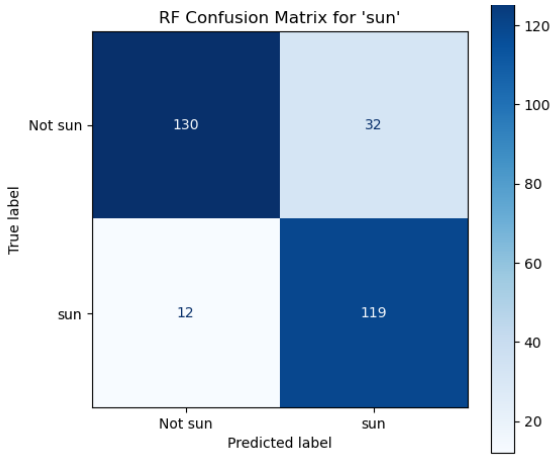

Figure 21. Confusion matrix for sun prediction using RF.

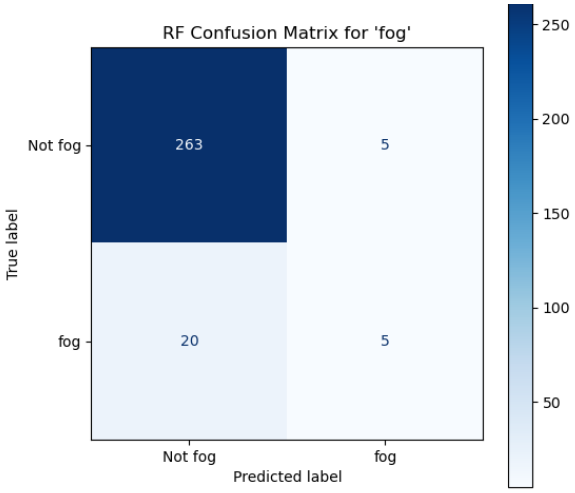

Figure 22. Confusion matrix for fog prediction using RF.

5. Conclusion

This project highlights the effective application of deep learning techniques, specifically Recurrent Neural Networks and Feed-Forward Neural Networks, in weather forecasting. The RNN model, designed for time-series regression, consisted of two SimpleRNN layers with Dropout regularization and a dense output layer. It processed an input of sequences of seven consecutive days to predict the next temperature ranges for the Seattle dataset and temperature values for the Jena dataset, in the end it was able to achieve a high accuracy for the Jena dataset, which was of better quality, and reasonable accuracy for the Seattle dataset. By employing the Mean Squared Error (MSE) loss function, our RNN model was able to penalize larger errors and be more sensitive to outlier data, it resulted in an effective ability to handle temporal dependencies and produce reliable predictions.

The FFNN model was tailored for structured data classification, which featured two hidden layers with Dropout and a softmax output layer. As shown in our results, it was able to classify weather types from the given dataset, and achieved an accuracy of 88%. While the dataset had different variations of minority and majority classes in terms of weather categories, our model was still able to perform well which can reflect on its ability to adapt to realistic weather patterns. Additionally, the added evaluation metrics such as precision, recall, F1-score, and accuracy helped us evaluate our FFNN model's performance, and observe the impact of the imbalanced Seattle dataset on its performance.

Overall, this project highlights the importance of dataset analysis and data augmentation when it comes to preparing data for machine learning or deep learning tasks. Challenges like dataset imbalances and feature relevance were

accounted for to better improve our model's performance. The final results of this project demonstrate the potential that deep learning techniques, such as RNNs and FFNNs, have for weather prediction. With more exploration, future plans could include real-time data integration with an easily deployable hardware device, hyper-parameter and model architecture optimization, and the engineering of even more features to improve both accuracy and scalability of both deep learning models.

References

[1] M. AbdulRaheem, J. B. Awotunde, A. E. Adeniyi, I. D. Oladipo, and S. O. Adekola, "Weather prediction performance evaluation on selected machine learning algorithms," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 11, no. 4, pp. 1535–1544, Dec. 2022, Available: https://ijai.iaescore.com/index.php/IJAI/article/view/21737/13496

[2] C. Z. Basha, N. Bhavana, P. Bhavya, and S. V, "Rainfall Prediction using Machine Learning & Deep Learning Techniques," IEEE Xplore, Jul. 01, 2020. https://ieeexplore.ieee.org/abstract/document/9155896

[3] N. Singh, S. Chaturvedi, and S. Akhter, "Weather Forecasting Using Machine Learning Algorithm," 2019 International Conference on Signal Processing and Communication (ICSC), Mar. 2019. https://ieeexplore.ieee.org/abstract/document/8938211

[4] Suri Babu Nuthalapati and Aravind Nuthalapati, "Accurate weather forecasting with dominant gradient boosting using machine learning," International Journal of Science and Research Archive, vol. 12, no. 2, pp. 408–422, Jul. 2024. https://www.researchgate.net/profile/Suri-Babu-Nuthalapati/publication/382636959_Accurate_weather_forecasting_with_dominant_gradient_boosting_using_machine_learning

[5] Googleapis.com, 2016. https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip

[6] "Exploring Seattle Weather — Vega-Altair 5.5.0 documentation," Github.io, 2015. https://altair-viz.github.io/case_studies/exploring-weather.html (accessed Dec. 09, 2024).

[7] Time series forecasting, "Time series forecasting — TensorFlow Core," TensorFlow, 2009. https://www.tensorflow.org/tutorials/structured_data/time_series

[8] R. Tatman, "Did it rain in Seattle? (1948-2017)," Kaggle.com, 2017. https://www.kaggle.com/datasets/rtatman/did-it-rain-in-seattle-19482017/data

[9] AbdElRahman16, "Weather Seattle," Kaggle.com, 2024. https://www.kaggle.com/datasets/abdelrahman16/weather-seattle/data

[10] M. Nassri, "Jena Climate Dataset (2009–2016)," Kaggle, 2017. https://www.kaggle.com/datasets/mnassrib/jena-climate