ClimateWins – Exercise 2.2: Deep Learning Model Summary

1. CNN Model Summary

The Convolutional Neural Network (CNN) model was built to classify pleasant weather patterns across 15 weather stations using a 1D temporal structure.

Key steps:

- Data reshaped to (22950, 15, 9) for input and (22950, 15) for labels

- Used Conv1D → MaxPooling1D → Dense(64) → Dense(15, softmax) architecture

- Trained for 10 epochs with batch size 16

- Results:

  - Training Accuracy: ~64.7%

  - Validation Accuracy: ~64.0%

  - Test Accuracy: ~63.5%

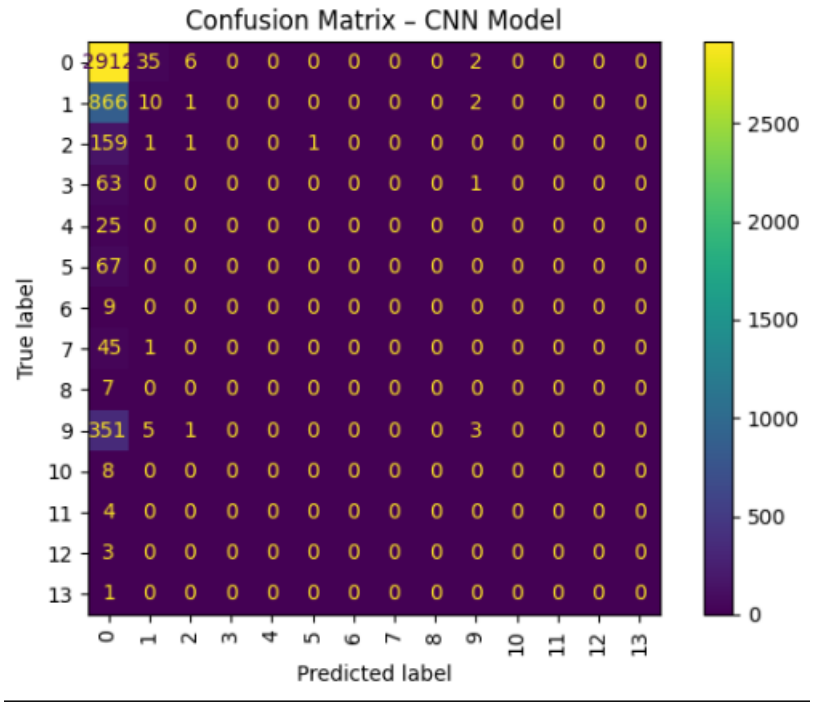

Fig: CNN Model

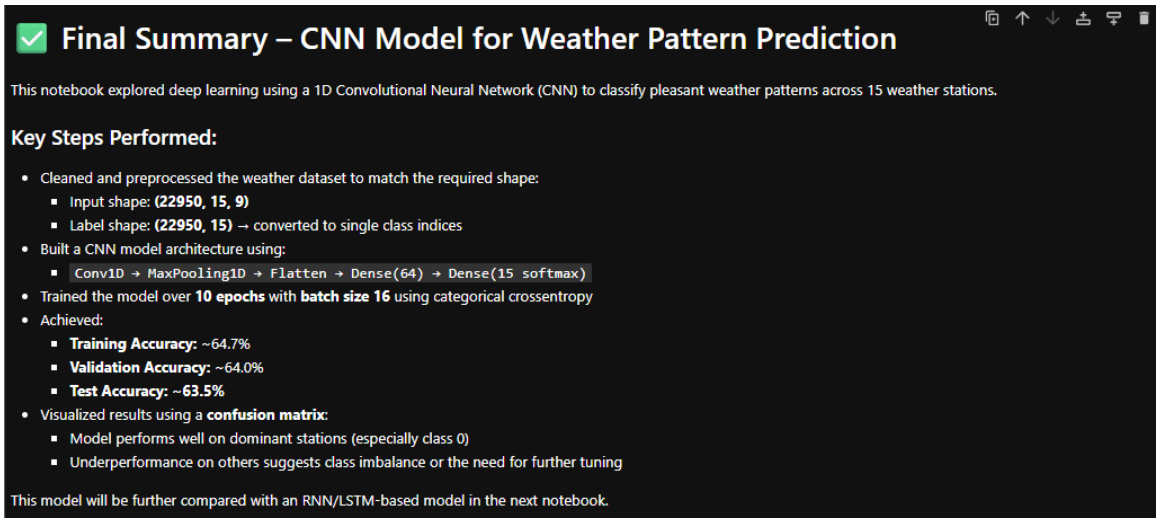

Fig: CNN confusion matrix

What the CNN Model Results Mean(simplified explanation)

In this project, we trained a computer model to recognize patterns in weather data and guess which weather station the data came from. The model looked at past weather records — things like temperature, wind, and humidity — and learned how these change over time. Each record was divided into small time blocks, kind of like watching 15 short clips to understand a full story. The model went through this process 10 times to learn as much as it could.

After training, we tested how well the model worked. It correctly guessed the right station about **65% of the time** — both while learning and when it was tested on new examples. That means it did pretty well, especially since it had to choose from 15 different stations. The confusion matrix (a table that shows which stations it got right or wrong) showed that the model was great at predicting some stations but had trouble with others. This likely happened because some stations had more data than others, making them easier to learn.

Overall, the model did a solid job. With a bit more data or by using a different type of model, we could likely make it even better in the future.

Interpreting the CNN Model Results(Technical Explanation)

The CNN model was trained to recognize pleasant weather patterns across 15 different weather stations using historical climate data. The model learned from over 22,000 examples, each broken into sequences of 15 time steps with 9 different weather-related

features per step. It was trained for 10 rounds (called epochs), and during this process, it became better at predicting which station a weather pattern belonged to. The results show that the model reached a **training accuracy of around 64.7%**, meaning it correctly classified nearly two-thirds of the examples it saw during training. Its **validation accuracy** (how well it performed on new, unseen data during training) was about **64.0%**, indicating the model is generalizing reasonably well. On the final **test dataset**, it achieved an accuracy of **63.5%**, showing consistent performance. The confusion matrix further reveals that the model performed especially well on certain stations (like class 0), but struggled with others, likely due to an imbalance in how frequently each station appeared in the dataset. Overall, the CNN model demonstrates promising ability to detect spatial weather patterns, with room for improvement using more balanced data or advanced techniques like RNNs or LSTMs in future exercises.

2. RNN Model Summary

A SimpleRNN-based model was implemented to capture sequential weather patterns from the same dataset.

Key steps:

- Input data shaped to (22950, 15, 9), labels remained at (22950, 15)

- Labels transformed into single-class indices and one-hot encoded

- Used SimpleRNN(64) → Dense(15, softmax) architecture

- Trained for 10 epochs with batch size 16

- Results:

- Training Accuracy: ~64.7%

- Validation Accuracy: ~64.6%

- Test Accuracy: ~64.3%

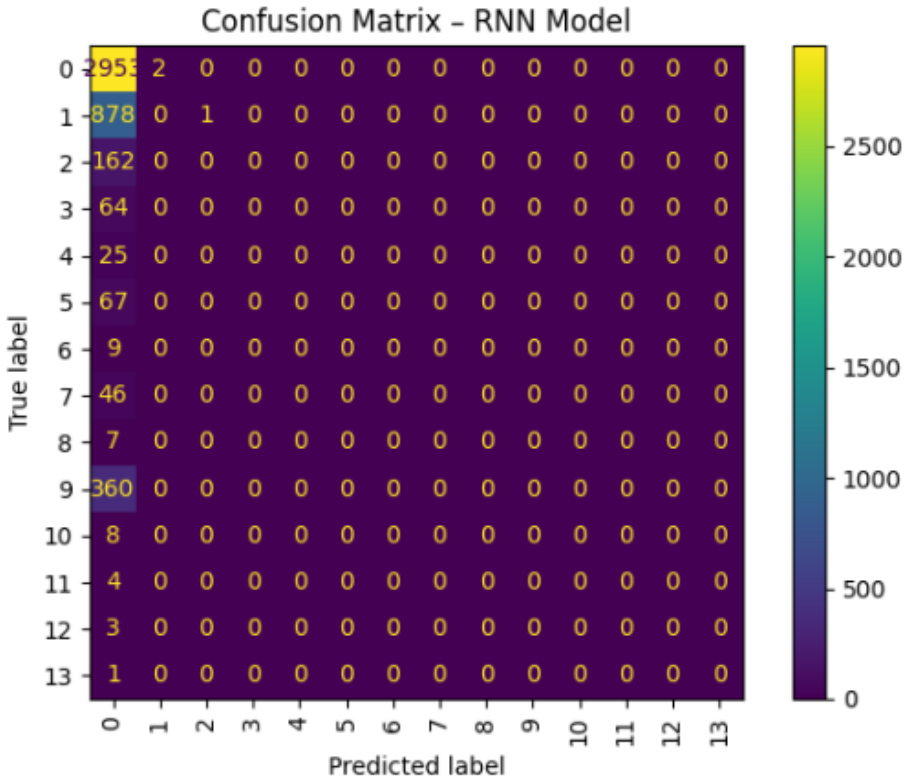

Fig: RNN confusion matrix

✅ Final Summary – RNN Model for Weather Pattern Prediction

This notebook implemented a Recurrent Neural Network (RNN) using SimpleRNN layers to classify weather station patterns over time.

Key Steps Performed:

- Loaded and reshaped weather data to match RNN input format: **(22950, 15, 9)**
- Converted label matrix to class indices and applied one-hot encoding
- Built an RNN model with the following architecture:
  - `SimpleRNN (64 units) → Dense (15 classes, softmax)`
- Trained for **10 epochs** with **batch size 16**, using categorical crossentropy
- Achieved:
  - **Training Accuracy:** ~64.7%

Fig: RNN Model Summary

What the RNN Model Results Mean (simplified explanation)

In this part of the project, we used a different type of model called an RNN (Recurrent Neural Network). It's designed to understand how things change over time, making it a good fit for weather data. Just like before, we gave the model lots of past weather records, split into 15 time blocks. The RNN looked at things like temperature, humidity, and wind — and tried to learn which weather station each pattern came from.

After training the RNN model 10 times (epochs), we tested how well it performed. It correctly guessed the station about **64% of the time**, which is very close to what the CNN model achieved. The confusion matrix showed the RNN also did well with some stations (especially class 0), but struggled with others — most likely because there weren't enough examples from those stations. Still, it showed solid overall performance.

RNNs are known for handling sequences well, so this result shows that it could capture the time-based nature of the weather data. With more balanced data or more advanced RNN types, we could probably improve accuracy even more in future experiments.

Interpreting the RNN Model Results (technical explanation)

The RNN model was applied to the same weather dataset, with input reshaped to sequences of 15 time steps and 9 features per step. This structure allows the model to learn temporal relationships across weather patterns. After aligning the input and output data to 22,950 samples and 15 output classes, the RNN was trained for 10 epochs with a

batch size of 16. The model architecture included a SimpleRNN layer with 64 units followed by a Dense(15, softmax) output layer.

During training, the model achieved a **training accuracy of ~64.7%**, **validation accuracy of ~64.6%**, and a **final test accuracy of ~64.3%** — slightly better than the CNN in this setup. The confusion matrix indicates strong performance in classifying frequent stations, particularly class 0, but shows weakness in less-represented stations. This suggests that, like the CNN, the RNN was influenced by class imbalance in the dataset.

Overall, the RNN model proved effective at identifying patterns over time, confirming its suitability for sequence-based climate data. Its performance consistency compared to the CNN makes it a valuable alternative, especially when further tuned or extended into LSTM-based architectures in future steps.

3. Conclusion

Both the CNN and RNN models demonstrated consistent performance, with RNN slightly outperforming CNN. However, both models struggled to generalize across all 15 weather stations, likely due to class imbalance. Further improvements could be made by exploring LSTM layers, data augmentation, or class weighting strategies.