Exercise 2.2 – Deep Learning With ClimateWins Weather Data

CNN Model for Pleasant Weather Prediction

1. Purpose of the Analysis

The goal of this exercise was to apply deep learning techniques to ClimateWins' long-term European weather data in order to identify large-scale weather patterns across multiple stations simultaneously. Specifically, a Convolutional Neural Network (CNN) was developed to predict pleasant weather conditions using unscaled historical weather observations collected over approximately 60 years.

This task focused on:

- Preparing multivariate time-series weather data for deep learning,
- Structuring the data correctly for a CNN model,
- Training and evaluating the model's ability to recognize weather patterns across 15 stations.

2. Data Sets Used

Two unscaled datasets were used:

- Weather Observations (X)
    - File: Dataset-weather-prediction-dataset-processed.csv
    - Contains weather observations for multiple European stations
- Pleasant Weather Labels (y)
    - File: Dataset-Answers-Weather_Prediction_Pleasant_Weather.csv
    - Binary indicators for whether weather conditions were classified as "pleasant" at each station

Both datasets were loaded directly into Python using Pandas.

3. Data Cleaning and Preparation

Column Removal

To align with task requirements:

- DATE and MONTH were removed from the observations dataset
- DATE was removed from the labels dataset

Station Consistency

- All observation columns were filtered to include only stations present in the labels dataset
- This ensured that every station had a complete set of labels

Handling Missing and Sparse Data

- Two observation types (snow_depth and wind_speed) were removed because they were missing multiple years of data across most stations
- Three individual missing station observations were filled by copying data from nearby geographic stations:
    - Ljubljana → Kassel
    - Sonnblick → München
    - Oslo → Stockholm

This approach follows the assumption that nearby stations experience similar weather patterns.

Final Cleaned Data Shapes

- X shape: (22950, 132)
- y shape: (22950, 15)

4. Structuring Data for Deep Learning

After cleaning, the dataset was reshaped to meet CNN input requirements.

Observation Grouping

- Only observation types present across all 15 stations were retained:
    - precipitation
    - temp_min

    - global_radiation
    - temp_mean
    - sunshine
    - temp_max

This resulted in 6 observation types per station.

Final Deep Learning Shapes

- X reshaped to: (22950, 15, 6)
    - 15 stations × 6 observation types
- y retained as: (22950, 15)
    - One label per station

This structure allows the CNN to learn both station-level and cross-station weather patterns.

5. Train–Test Split

The reshaped data was split into training and testing sets:

- Training set: 80%
- Testing set: 20%

Resulting Shapes

- X_train: (18360, 15, 6)
- X_test: (4590, 15, 6)
- y_train: (18360, 15)
- y_test: (4590, 15)

6. Model Selection and Justification

Why a CNN Was Chosen

A Convolutional Neural Network (CNN) was selected instead of an RNN because:

- CNNs are highly effective at learning spatial relationships, which aligns well with multi-station weather data
- The observation structure (stations × features) is better treated as a grid of related signals rather than a purely temporal sequence
- CNNs train faster and are less computationally expensive than RNNs for this type of structured numerical data

7. CNN Model Architecture

The CNN model was built using Keras with TensorFlow, following the HAR sample structure.

Key Hyperparameters

- Hidden layers increased in powers of 2
- Small initial values were used to prevent overfitting
- ReLU activation used in convolutional layers
- Sigmoid activation used in the final dense layer
- Adam optimizer
- Binary cross-entropy loss function

Screenshot 1 – Initial CNN Model Summary

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 13, 16)	304
max_pooling1d (MaxPooling1D)	(None, 6, 16)	0
conv1d_1 (Conv1D)	(None, 4, 32)	1,568
max_pooling1d_1 (MaxPooling1D)	(None, 2, 32)	0
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 64)	4,160
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 15)	975

Total params: 7,007 (27.37 KB)

Trainable params: 7,007 (27.37 KB)

Non-trainable params: 0 (0.00 B)

8. Model Training and Tuning

The model was trained while monitoring:

- Training loss
- Validation accuracy

Hyperparameters were adjusted iteratively to reduce loss and improve generalization without overfitting.

Screenshot 2 – Final CNN Model Summary

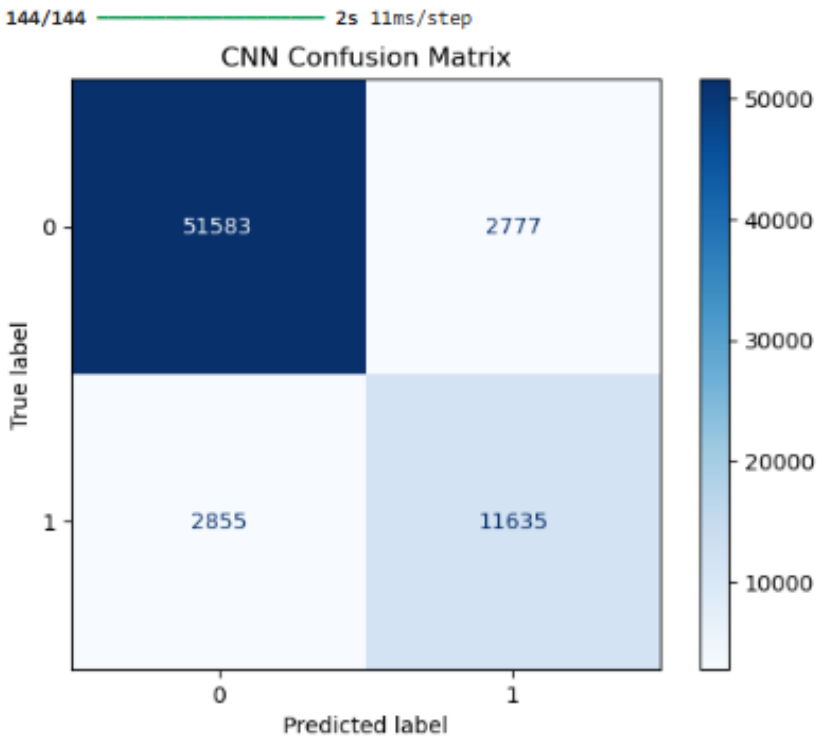
```
Epoch 1/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 13s 14ms/step - accuracy: 0.1145 - loss: 0.3244 - val_accuracy: 0.1430 - val_loss: 0.2511
Epoch 2/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1454 - loss: 0.2577 - val_accuracy: 0.1629 - val_loss: 0.2290
Epoch 3/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1614 - loss: 0.2412 - val_accuracy: 0.1789 - val_loss: 0.2206
Epoch 4/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1642 - loss: 0.2312 - val_accuracy: 0.1678 - val_loss: 0.2191
Epoch 5/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1705 - loss: 0.2244 - val_accuracy: 0.1732 - val_loss: 0.2029
Epoch 6/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1714 - loss: 0.2178 - val_accuracy: 0.1729 - val_loss: 0.1988
Epoch 7/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1725 - loss: 0.2141 - val_accuracy: 0.1901 - val_loss: 0.1945
Epoch 8/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.1799 - loss: 0.2095 - val_accuracy: 0.1678 - val_loss: 0.1907
Epoch 9/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.1828 - loss: 0.2053 - val_accuracy: 0.1865 - val_loss: 0.1866
Epoch 10/10
459/459 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.1871 - loss: 0.2018 - val_accuracy: 0.1885 - val_loss: 0.1836
```

9. Model Evaluation

Confusion Matrix

A confusion matrix was generated to visualize prediction performance across stations.

Screenshot 3 – Final Confusion Matrix



10. Accuracy Results

Overall Accuracy

- Overall label accuracy across all stations: 91.82%

Per-Station Accuracy

- Station 01: 88.39%
- Station 02: 88.39%
- Station 03: 88.93%
- Station 04: 91.31%
- Station 05: 91.72%
- Station 06: 88.76%
- Station 07: 92.92%
- Station 08: 90.72%
- Station 09: 91.85%
- Station 10: 91.18%
- Station 11: 92.33%
- Station 12: 93.53%
- Station 13: 100%
- Station 14: 92.27%
- Station 15: 95.01%

The model successfully recognized all 15 stations, meeting the task requirement.

Screenshot 4 – Accuracy Output / Metrics

```
Overall label accuracy across all stations: 0.9181989832970225
Station 01 accuracy: 0.8838779956427015
Station 02 accuracy: 0.8838779956427015
Station 03 accuracy: 0.8893246187363835
Station 04 accuracy: 0.9130718954248366
Station 05 accuracy: 0.9172113289760349
Station 06 accuracy: 0.8875816993464052
Station 07 accuracy: 0.9291938997821351
Station 08 accuracy: 0.9071895424836601
Station 09 accuracy: 0.9185185185185185
Station 10 accuracy: 0.9117647058823529
Station 11 accuracy: 0.9233115468409586
Station 12 accuracy: 0.9352941176470588
Station 13 accuracy: 1.0
Station 14 accuracy: 0.9226579520697168
Station 15 accuracy: 0.9501089324618737
```

11. Conclusion

This exercise successfully demonstrated how deep learning models can be applied to long-term, multi-station climate data. The CNN model effectively learned complex spatial weather patterns and achieved strong predictive performance across all stations.