UNSUPERVISED

MACHINE LEARNING WEATHER PREDICTION ANALYSIS

ING. LUIS A. GIL LARES

NOVEMBER 2024

Project Overview

- **Objective:** Identify critical features of climate data to predict pleasant weather and improve resource allocation.
- **Tools & Methods:** Python, Random Forests, Neural Networks, Hyperparameter Optimization, Visualizations.
- **Key Question:** What data features best predict pleasant weather?

Climate refugees are a growing reason for emigration globally, including in Europe.
Source: Steve Evans (CC BY-NC 2.0)

Dataset Overview

- **Weather Data:**
  - 170 features from multiple weather stations (temperature, humidity, precipitation, etc.).
- **Pleasant Weather Labels:**
  - Binary labels for weather conditions by station.
- Data cleaned and structured to ensure balance.

In [3]:
```python
# Import the orders data
weather_data = pd.read_csv(os.path.join(path, 'Data Sets', 'Weather.csv'))
weather_data
```

Out[3]:

	DATE	MONTH	BASEL_cloud_cover	BASEL_wind_speed	BASEL_humidity	BASEL_pressure	BASEL_global_radiation	BASEL_precipitation	BASEL_s
0	19600101	1	7	2.1	0.85	1.0180	0.32	0.09	
1	19600102	1	6	2.1	0.84	1.0180	0.36	1.05	
2	19600103	1	8	2.1	0.90	1.0180	0.18	0.30	
3	19600104	1	3	2.1	0.92	1.0180	0.58	0.00	
4	19600105	1	6	2.1	0.95	1.0180	0.65	0.14	
...	...	...	...	...	...	...	...	...	
22945	20221027	10	1	2.1	0.79	1.0248	1.34	0.22	
22946	20221028	10	6	2.1	0.77	1.0244	1.34	0.22	
22947	20221029	10	4	2.1	0.76	1.0227	1.34	0.22	
22948	20221030	10	5	2.1	0.80	1.0212	1.34	0.22	
22949	20221031	10	5	2.1	0.84	1.0193	1.34	0.22	

22950 rows × 170 columns

In [4]:
```python
# Import the pleasant weather data
pleasant_weather = pd.read_csv(os.path.join(path, 'Data Sets', 'Dataset-Answers-Weather_Prediction_Pleasant_Weather.csv'))
pleasant_weather
```

Out[4]:

	DATE	BASEL_pleasant_weather	BELGRADE_pleasant_weather	BUDAPEST_pleasant_weather	DEBILT_pleasant_weather	DUSSELDORF_pleasant_weath
0	19600101	0	0	0	0	
1	19600102	0	0	0	0	
2	19600103	0	0	0	0	
3	19600104	0	0	0	0	
4	19600105	0	0	0	0	
...	...	...	...	...	...	
22945	20221027	0	0	0	0	
22946	20221028	0	0	0	0	
22947	20221029	0	0	0	0	
22948	20221030	0	0	0	0	
22949	20221031	0	0	0	0	

22950 rows × 16 columns

Data Cleaning and RNN Model Development

- **Overview**
- **Objective**: Classify pleasant weather conditions for European cities using RNNs.
- **Dataset**:
  - Original: 22,950 samples, 170 features.
  - Cleaned: 134 features, (22,950, 15, 9) shape for the model.
- **Data Cleaning**

1. **Removed**:
   1. Columns for Gdansk, Roma, Tours, wind_speed, and snow_depth.
2. **Added**:
   1. Data for Kassel, Sonnblick, Oslo, using nearby stations.
3. **Final Dataset**:
   1. Features: Temp_mean, cloud_cover, humidity, pressure, etc.

- **Data Splits**
- **Training Data**: (18,360, 15, 9)
- **Testing Data**: (4,590, 15, 9)

```
In [8]: # List of cities to be removed
cities_to_remove = ['GDANSK', 'ROMA', 'TOURS']

# Filter out columns that contain the names of the cities to remove
columns_to_keep = [col for col in weather_data.columns if not any(city in col for city in cities_to_remove)]

# Create a new dataframe with the filtered columns
weather_data_filtered = weather_data[columns_to_keep]

# Check the shape to ensure the columns were removed
print(f"Shape of weather_data_filtered: {weather_data_filtered.shape}")

Shape of weather_data_filtered: (22950, 149)
```

```
In [9]: # Assuming X is 2D array or DataFrame with shape (22950*15, 9)


X = X.reshape(-1, 15, 9)
print("Reshaped X:", X.shape)  # Should output (22950, 15, 9)

Reshaped X: (22950, 15, 9)
```

RNN Model Reports and Insights

In [18]:
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
144/144 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step
Pred          BASEL  BUDAPEST  MAASTRICHT  MUNCHENB
True
BASEL           270      2592          26        21
BELGRADE        220       681           7         1
BUDAPEST         14       167           0         0
DEBILT           12        59           0         0
DUSSELDORF        1        25           0         0
HEATHROW          3        83           0         0
KASSEL            0         6           0         0
LJUBLJANA         1        48           0         0
MAASTRICHT        0         4           0         0
MADRID            8       323           0         0
MUNCHENB          1         5           0         0
OSLO              0         7           0         0
STOCKHOLM         0         2           0         0
VALENTIA          0         3           0         0
```

CNN + tanh (30 epochs, 128 LSTM units

- Validation Accuracy: 9.52% (final).Issue:
- High confusion; BASEL misclassified as BUDAPEST.

In [22]:
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
144/144 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step
Pred          BASEL
True
BASEL          2909
BELGRADE        909
BUDAPEST        181
DEBILT           71
DUSSELDORF       26
HEATHROW         86
KASSEL            6
LJUBLJANA        49
MAASTRICHT        4
MADRID          331
MUNCHENB          6
OSLO              7
STOCKHOLM         2
VALENTIA          3
```

CNN + sigmoid (20 epochs, 128 LSTM units)

- Validation Accuracy: 3.70%.
- Issue: Predicted BASEL for most classes.

In [26]:
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
144/144 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step
Pred          MAASTRICHT  OSLO
True
BASEL                  1  2908
BELGRADE               0   909
BUDAPEST               0   181
DEBILT                 0    71
DUSSELDORF             0    26
HEATHROW               0    86
KASSEL                 0     6
LJUBLJANA              0    49
MAASTRICHT             0     4
MADRID                 0   331
MUNCHENB               0     6
OSLO                   0     7
STOCKHOLM              0     2
VALENTIA               0     3
```

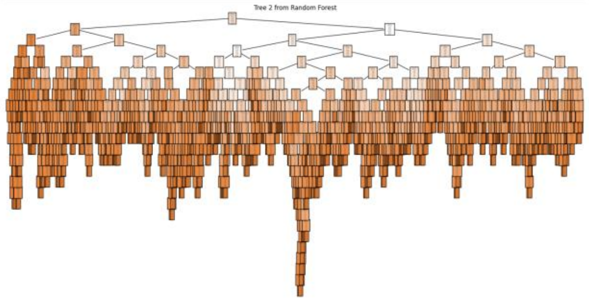
CNN + tanh (20 epochs, 128 LSTM units)

- Validation Accuracy: 0.15%.
- Issue: Strong bias toward OSLO.

Data Preparation, Feature Selection & Modeling

- **1. Feature Selection**
  - **Target Variable:** Binary classification (0 = Non-pleasant, 1 = Pleasant).
  - **Key Predictors:** Temperature-related metrics emerged as the most significant features.
- **2. Modeling**
  - **Algorithm:** Random Forest Classifier for robustness and feature handling.
  - **Train-Test Split:** Data split for unbiased evaluation on unseen data.
- **3. Model Evaluation**
  - **Accuracy:** Achieved 57.59%, indicating moderate success with room for improvement.
  - **Classification Report Highlights:**
    - **BASEL & BUDAPEST:** Strong performance.
    - **SONNBLICK & VALENTIA:** Struggled due to class imbalance and underrepresentation.

```
In [30]: # Plot the second tree
plt.figure(figsize=(20,10))
plot_tree(tree2, filled=True, rounded=True, feature_names=X.columns, class_names=['Not Pleasant', 'Pleasant'])
plt.title("Tree 2 from Random Forest")
plt.show()
```

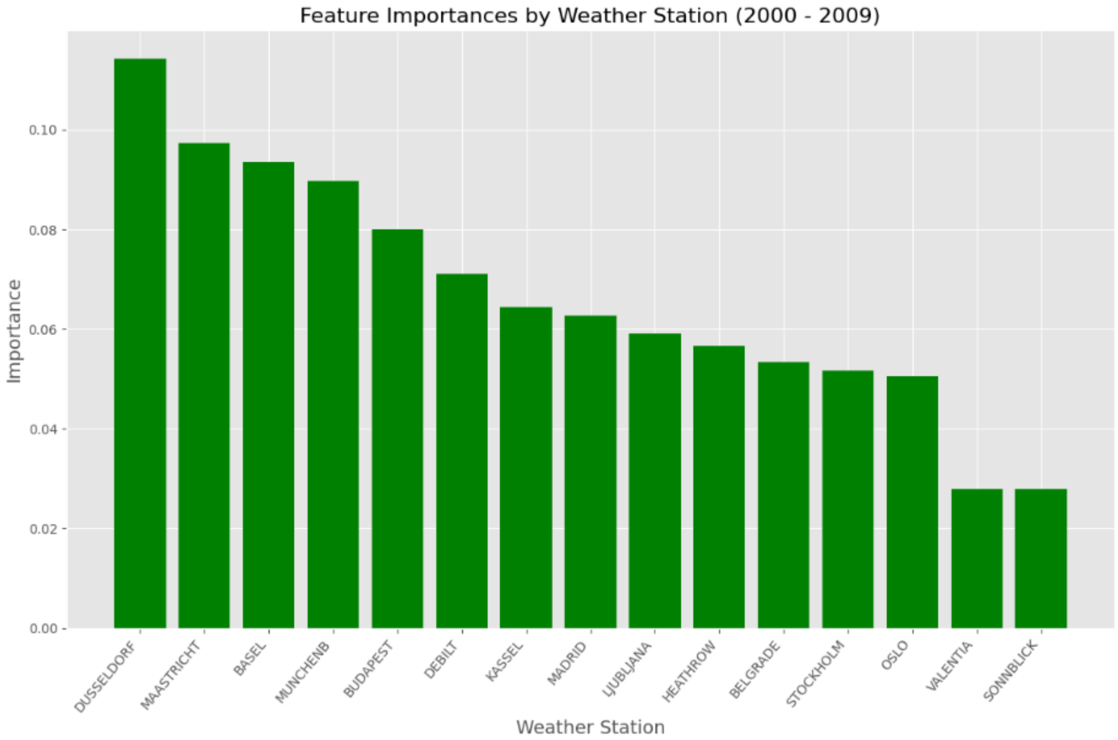



```
In [25]: # Train the model
rf_model.fit(X_train, y_train)

# Predict on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
print(classification_report(y_test, y_pred))
```

Accuracy: 0.5759

	precision	recall	f1-score	support
0	0.96	0.91	0.93	190
1	0.89	0.93	0.91	270
2	0.87	0.98	0.92	255
3	0.92	0.90	0.91	155
4	0.93	0.87	0.90	158
5	0.88	0.73	0.80	160
6	0.93	0.78	0.85	147
7	0.87	0.90	0.88	209
8	0.97	0.91	0.94	160
9	0.91	0.99	0.94	348
10	0.96	0.87	0.91	169
11	0.93	0.70	0.80	120
12	0.00	0.00	0.00	0
13	0.90	0.75	0.82	127
14	1.00	0.05	0.09	44
micro avg	0.91	0.87	0.89	2512
macro avg	0.86	0.75	0.77	2512
weighted avg	0.91	0.87	0.88	2512
samples avg	0.55	0.51	0.52	2512

Data Preparation, Feature Selection & Modeling

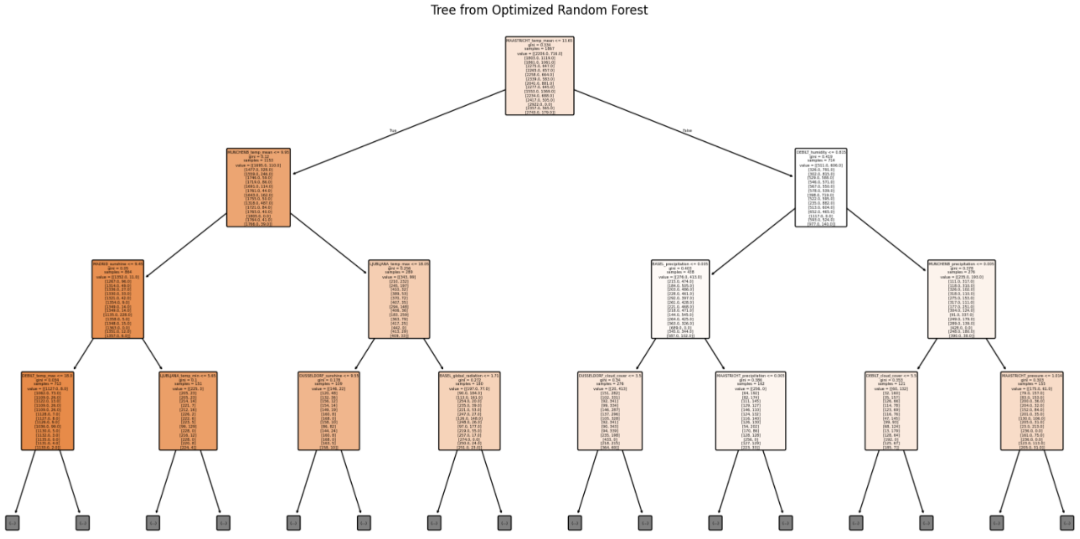


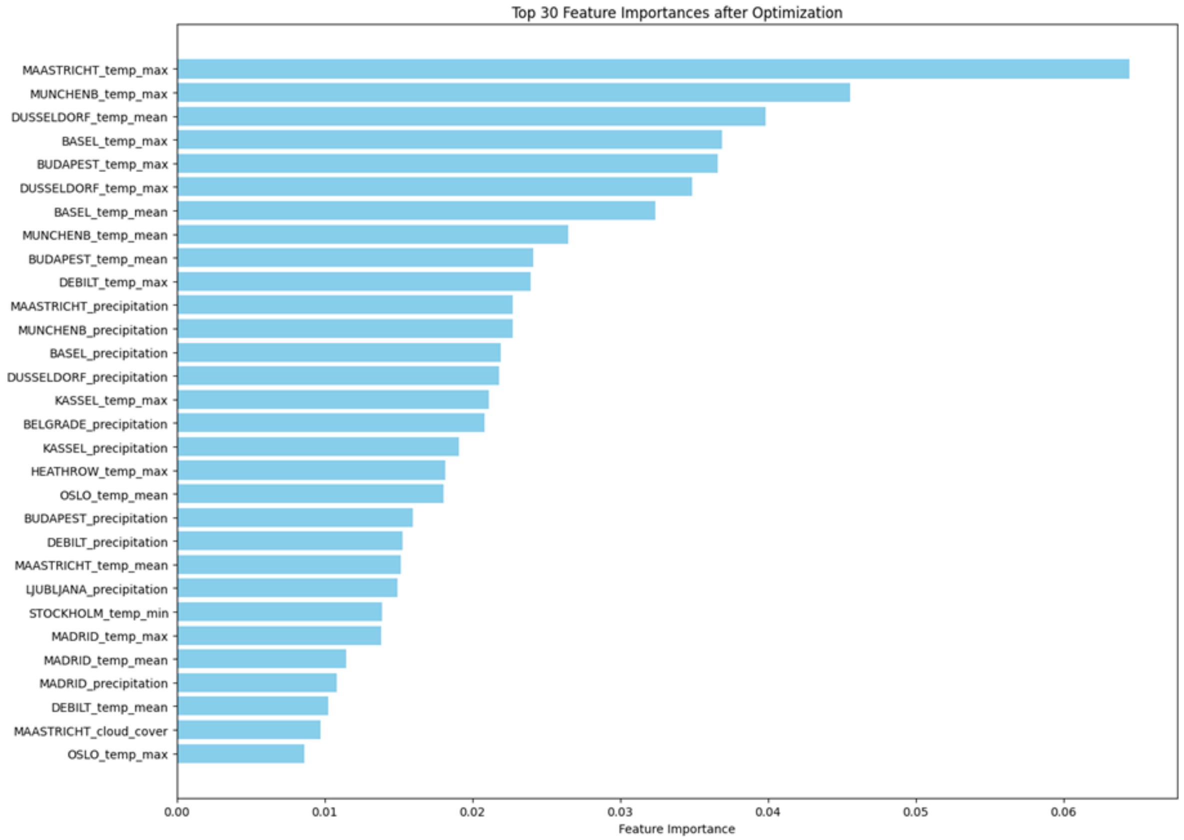
- **5. Key Stations with Perfect Accuracy (1.0000):**
- **DUSSELDORF, MAASTRICHT, BASEL**
  - **Top Features:** Precipitation, max temperature, mean temperature, sunshine, global radiation, humidity, and cloud cover.
  - **Metrics:** Perfect precision, recall, and F1-scores across all classes.
- **6. Challenges & Issues**
- **Class Imbalance:** Limited data for certain stations like SONNBLICK and VALENTIA.
- **Performance Variability:** Some stations show underwhelming results due to data limitations.

Random Forest Optimization and Key Results

- **1. Summary of Results:**
- **Initial Model (Task 2.3):**
  - Accuracy: **0.5759**
  - Key Features: Predominantly temperature metrics and precipitation.
- **Optimized Model (Task 2.4):**
  - Accuracy: **0.5746** (using Randomized Search due to computational constraints).
  - Key Features: Consistent importance of temperature metrics (e.g., MAASTRICHT_temp_max, MUNCHENB_temp_max).
- **2. Feature Importance Analysis:**
- **Primary Variables:**
  - MAASTRICHT_temp_max, MUNCHENB_temp_max, and DUSSELDORF_temp_mean retained their importance.
- **Minor Adjustments:**
  - Post-optimization, cloud cover and sunshine metrics gained slightly more relevance.
- **3. Observations:**
- **Stability:** Limited changes in feature rankings highlight the robustness of temperature as a key predictor.
- **Performance:** Slight accuracy drop suggests interpretability improvement without significant prediction gains.

Random Forest Optimization and Key Results





Deep Learning Optimization and Iterative Approach

- **1. Bayesian Hyperparameter Optimization:**
- **Optimized Configuration:**
  - Activation: **Softsign**
  - Batch Size: **460**, Epochs: **47**, Neurons: **61**
  - Kernel Size: **2**, Layers: **3 (Dense with Dropout)**
  - Optimizer: **Adadelta** (Learning Rate: 0.7631).
- **2. Performance Results:**
- **Improved Accuracy:** Better classification for BASEL, BELGRADE, and MADRID stations.
  - BASEL: **3,512** correct classifications (significant improvement).
  - BELGRADE: **956**, MADRID: **369** correct classifications.
- **Overfitting Signs:** High learning rate and batch size might cause overfitting; further regularization needed.
- **Part 3: Iterative Approach**
  1. Data Segmentation:
- Weather station groupings (e.g., urban, coastal).
- Time-based splits (hourly, daily, seasonal).
- Focused critical variables: temperature, wind speed, visibility, and precipitation.

Deep Learning Optimization and Iterative Approach

```
In [29]: # Evaluate

         print(confusion_matrix(y_test, y_pred, stations))
```

```
Pred        BASEL  BELGRADE  BUDAPEST  DEBILT  DUSSELDORF  HEATHROW  \
True
BASEL        3512        82        16       4           7         1
BELGRADE      121       956         4       0           0         1
BUDAPEST       21        42       140       2           2         0
DEBILT         15         9        25      28           5         0
DUSSELDORF      6         2         2       2           8         1
HEATHROW       13         4         3       0           3        40
KASSEL          3         2         1       0           2         0
LJUBLJANA       8         5         5       0           0         0
MAASTRICHT      6         0         0       1           0         0
MADRID         39        22        12       0           1         7
MUNCHENB        7         0         0       0           0         0
OSLO            0         0         0       0           0         0
STOCKHOLM       2         0         1       0           0         0
VALENTIA        1         0         0       0           0         0

Pred        LJUBLJANA  MAASTRICHT  MADRID  OSLO
True
BASEL               8           2      50     0
BELGRADE            2           0       8     0
BUDAPEST            2           0       5     0
DEBILT              0           0       0     0
DUSSELDORF          2           0       6     0
HEATHROW            2           0      17     0
KASSEL              0           0       2     1
LJUBLJANA          33           0       9     1
MAASTRICHT          0           1       1     0
MADRID              8           0     369     0
MUNCHENB            1           0       0     0
OSLO                0           0       4     1
STOCKHOLM           0           0       1     0
VALENTIA            0           0       0     0
```

Handwriting Recognition Task Results

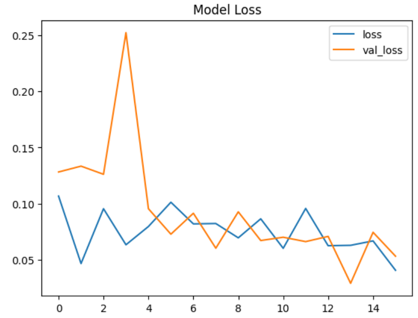
- **1. Model Performance:**
  - **Accuracy: 10.00%** (2/20 images correctly identified).
  - **Prediction Behavior:** Model predominantly predicted the number **8**, indicating overfitting or inadequate training.
- **2. Analysis of Results:**
  - **Correct Predictions:** Only images 8 and 18 (actual number 8).
  - **Misclassification Patterns:** Most numbers misclassified as **8** or unrelated digits.

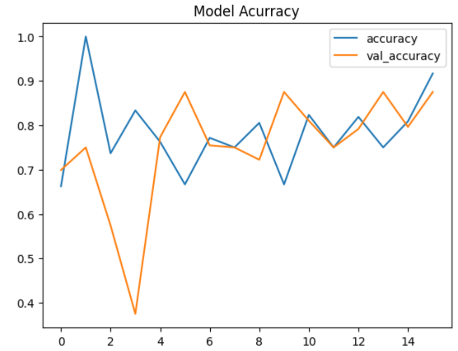
```
In [62]: from sklearn.metrics import accuracy_score
         # Calculate accuracy
         accuracy = accuracy_score(y_real_test, predicted_labels)
         print(f"Model accuracy on handwritten data: {accuracy * 100:.2f}%")

         Model accuracy on handwritten data: 10.00%
```

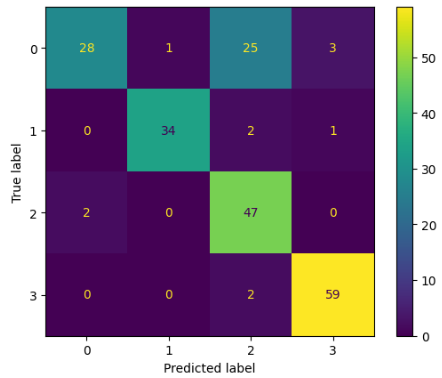
```
In [63]: # Display the predicted and true labels
         for i, (pred, actual) in enumerate(zip(predicted_labels, y_real_test)):
             print(f"Image {i}: Predicted = {pred}, Actual = {actual}")

         Image 0: Predicted = 8, Actual = 1
         Image 1: Predicted = 8, Actual = 10
         Image 2: Predicted = 8, Actual = 2
         Image 3: Predicted = 8, Actual = 3
         Image 4: Predicted = 0, Actual = 4
         Image 5: Predicted = 8, Actual = 5
         Image 6: Predicted = 0, Actual = 6
         Image 7: Predicted = 5, Actual = 7
         Image 8: Predicted = 8, Actual = 8
         Image 9: Predicted = 8, Actual = 9
         Image 10: Predicted = 8, Actual = 1
         Image 11: Predicted = 8, Actual = 10
         Image 12: Predicted = 7, Actual = 2
         Image 13: Predicted = 2, Actual = 3
         Image 14: Predicted = 3, Actual = 4
         Image 15: Predicted = 4, Actual = 5
         Image 16: Predicted = 3, Actual = 6
         Image 17: Predicted = 8, Actual = 7
         Image 18: Predicted = 8, Actual = 8
         Image 19: Predicted = 8, Actual = 9
```

Radar Recognition Task Results

- **1. Model Performance:**
  - **Training Accuracy: 91.67%, Validation Accuracy: 87.50%.**
  - **Training Loss: 0.041, Validation Loss: 0.053.**
- **2. Confusion Matrix Insights:**
  - Strong performance in predicting "Sunrise" with minimal misclassifications.
  - Frequent confusion between **"Cloudy"** and **"Shine."**
- **3. Observations:**
  - Model successfully classified weather types in most cases with high confidence for "Sunrise."







Radar Recognition Task Results

Correct Prediction - class: Sunrise - predicted: Sunrise[1.4477209e-13 5.6362812e-09 3.1840572e-10 1.0000000e+00]

Correct Prediction - class: Sunrise - predicted: Sunrise[1.0717241e-35 3.7137514e-22 1.7717912e-30 1.0000000e+00]

Correct Prediction - class: Rain - predicted: Rain[0.03280307 0.9378439 0.01369535 0.01565767]

Correct Prediction - class: Sunrise - predicted: Sunrise[6.56038102e-18 1.09890826e-10 2.66064772e-14 1.00000000e+00]

Correct Prediction - class: Sunrise - predicted: Sunrise[5.7174879e-24 9.1019090e-16 4.7867107e-19 1.0000000e+00]

Summary and Recommendations

- **Most Promising Thought Experiment:**
  GAN Applications for Weather Prediction
- **Potential:** Generate synthetic data for underrepresented weather conditions, visualize forecasts, and detect anomalies, enhancing real-world forecasting and anomaly detection.
- **Key Algorithms & Data Needed:**

1. **GAN Applications:**
   1. Algorithms: GANs, Conditional GANs.
   2. Data: Labeled weather images, meteorological data (e.g., temperature, wind speed).
2. **Handwriting Recognition:**
   1. Algorithms: CNNs, RNNs.
   2. Data: Augmented handwriting datasets, improved preprocessing.
3. **Radar Classification:**
   1. Algorithms: CNNs, Transfer Learning.
   2. Data: Radar images, synthetic data for balancing classes.

- **Next Steps:**
- **Focus on GANs**: Develop models to generate synthetic weather data.
- **Enhance Datasets**: Expand with diverse, labeled images and use augmentation.
- **Refine Models**: Conduct hyperparameter tuning and regularization for robustness.

luisgil1989@gmail.com

Thank you for your attention