Multi-Class Image Classification for Robust Outdoor Weather Recognition

PROJECT OVERVIEW

In the "Visual Weather Recognition with Deep Learning" project, the objective is to create a sophisticated Convolutional Neural Network (CNN) model capable of accurately classifying diverse weather conditions based on visual cues in images. By leveraging deep learning techniques, the project aims to develop a robust system that can distinguish between various weather phenomena, including sunny, rainy, cloudy, and sunrise scenes. The key components include the collection and exploration of a comprehensive dataset, data preprocessing with augmentation to enhance model generalization, and the design of a specialized CNN architecture. The model will be trained and evaluated, with a focus on metrics such as accuracy, precision, and recall. Visualization techniques will be employed to gain insights into the CNN's interpretation of different weather patterns, and considerations for potential deployment scenarios will be explored.

Technologies utilized in this project include Python as the primary programming language, TensorFlow for building and training the CNN, Matplotlib and Seaborn for visualization.

BUSINESS PROBLEM

The business objective is to leverage the multi-class weather dataset to advance the development of image classification models capable of accurately recognizing and categorizing different weather conditions. By doing so, this project aims to enhance the capabilities of outdoor weather analysis, offering valuable insights for sectors ranging from transportation and agriculture to urban planning and disaster preparedness.

PROJECT OBJECTIVES

The primary aim of this project is to develop an advanced Convolutional Neural Network (CNN) model for accurate and robust weather classification based on visual cues in images. Leveraging the power of deep learning, this project seeks to create a sophisticated system capable of distinguishing between various weather conditions, including sunny, rainy, cloudy, and sunrise scenes. The objective extends beyond mere image recognition to the creation of a model that exhibits high-level understanding of complex visual patterns associated with different weather phenomena.

* Achieve an accuracy of at least 90% in classifying outdoor weather images across various conditions.
* Develop a model that is robust to variations in lighting, camera angles, and weather intensity.
* Minimize computational resources required for model training and inference.

METRICS FOR SUCCESS

* Accuracy: Percentage of correctly classified weather images.
* Precision and Recall: Measures of how well the model identifies specific weather classes.
* F1-Score: Harmonic mean of precision and recall, providing a balanced measure of model performance.
* Inference Speed: Time taken to classify a single image.
* Computational Cost: Resources required to train and run the model.
* A robust and accurate multi-class image classification model for outdoor weather recognition.
* Potential applications in various sectors such as agriculture, aviation, and disaster management.

METHODS

The project will employ a rigorous data analysis process to understand the characteristics of the multi-class weather image dataset and prepare it for model development. Specifically, it will involve:

* Exploratory Data Analysis (EDA):
  * Analyze the distribution of images across weather classes to identify potential imbalances.
  * Explore the distribution of image sizes, resolutions, and color spaces to ensure consistency.
  * Visually analyze sample images from each class to identify key features and characteristics.
  * Employ data visualization techniques to gain insights into the data distribution and relationships between variables.
* Data Preprocessing:
  * Resizing and normalize all images to a consistent format compatible with the chosen CNN architecture.
  * Encoding class labels using one-hot encoding for multi-class classification.

METHODS cont...

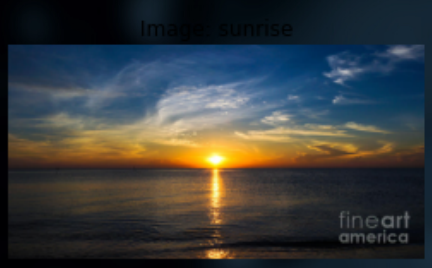
* Splitting the dataset into training, validation, and testing sets for model evaluation.
* Model Development:
 * Model Architecture:
 * Designing a multi-class image classification model using TensorFlow and Keras, considering the specific characteristics of the weather image dataset.
 * Exploring and experimenting with different CNN architectures, such as MobileNetV2, EfficientNetB3, ResNet, VGG, or Inception, to identify the most suitable architecture for this task.
 * Optimizing hyperparameters like learning rate, optimizer, and batch size to improve model performance.
 * Model Training and Evaluation:
 * Training the chosen model on the training set, monitoring its performance on the validation set to prevent overfitting.
 * Evaluating the model's performance on the unseen test set using metrics such as accuracy, precision, recall, and F1-score.

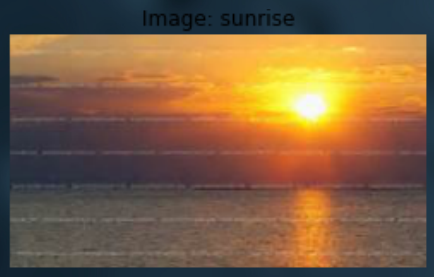
EDA BEFORE DATA PRE-PROCESSING

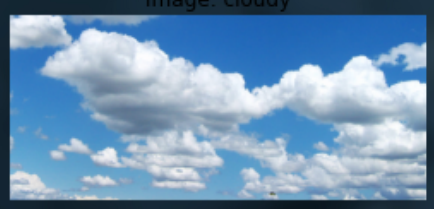
* This is to prepare the data in a format that is good to feed to the models. It involved the following steps:

1. Image Preview

Here, we had a look at 20 random images from the dataset:

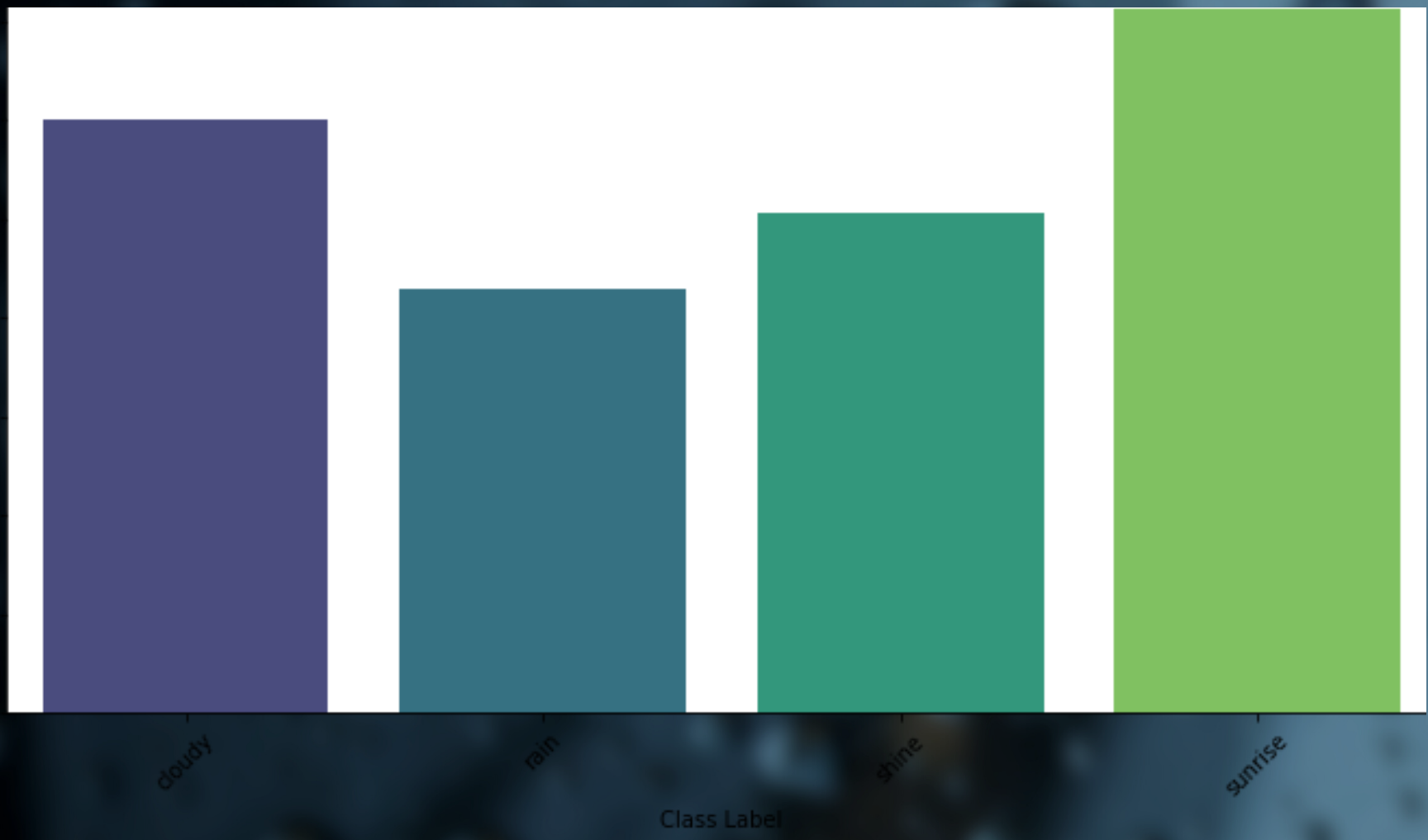






EDA BEFORE DATA PRE-PROCESSING

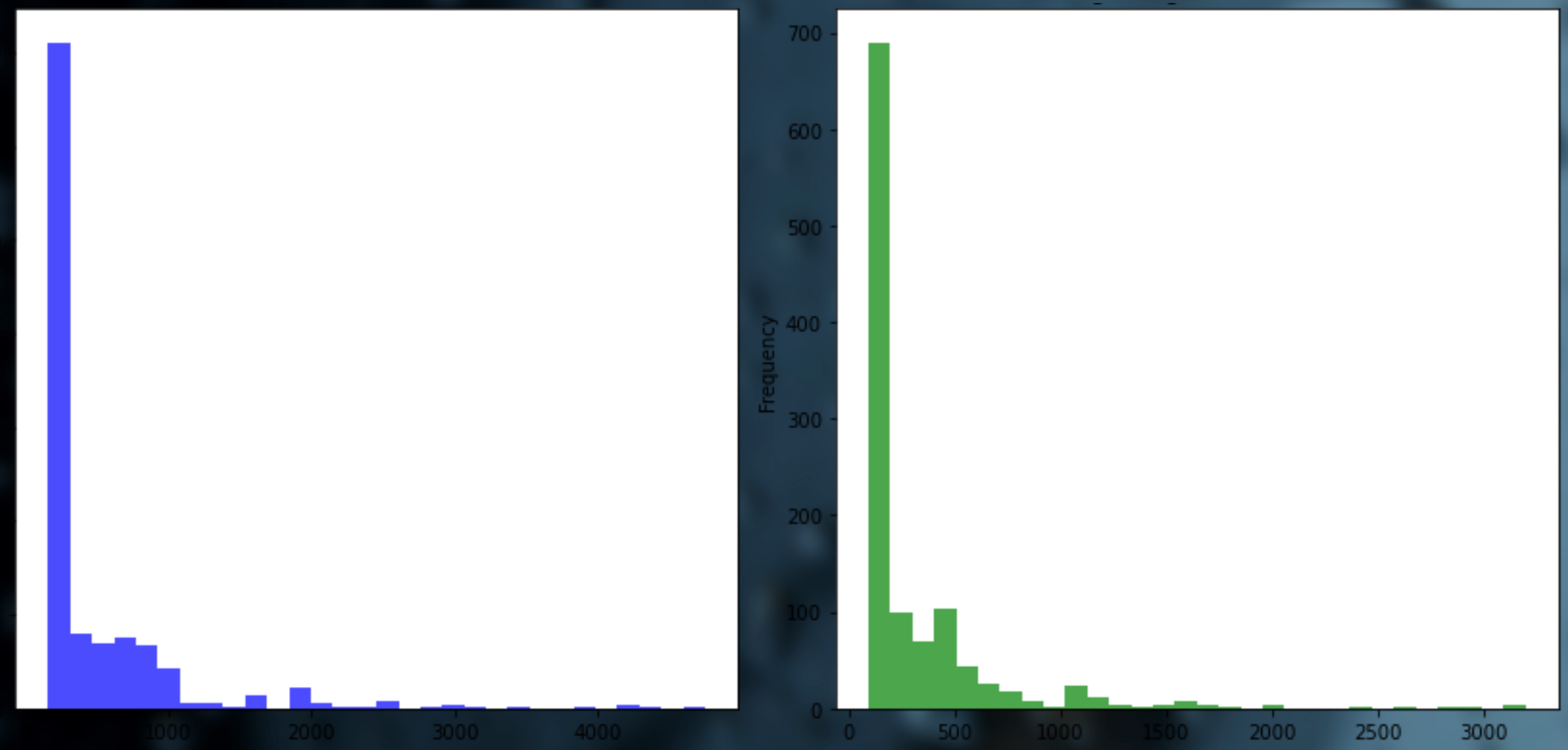
2. Weather class distriibution



Observation: The weather classes are relatively evenly distributed. There are 300 images showing cloudy weather, 215 images showing rainy weather, 253 images showing shinny weather, and 357 images showing sunrise.

EDA BEFORE DATA PRE-PROCESSING

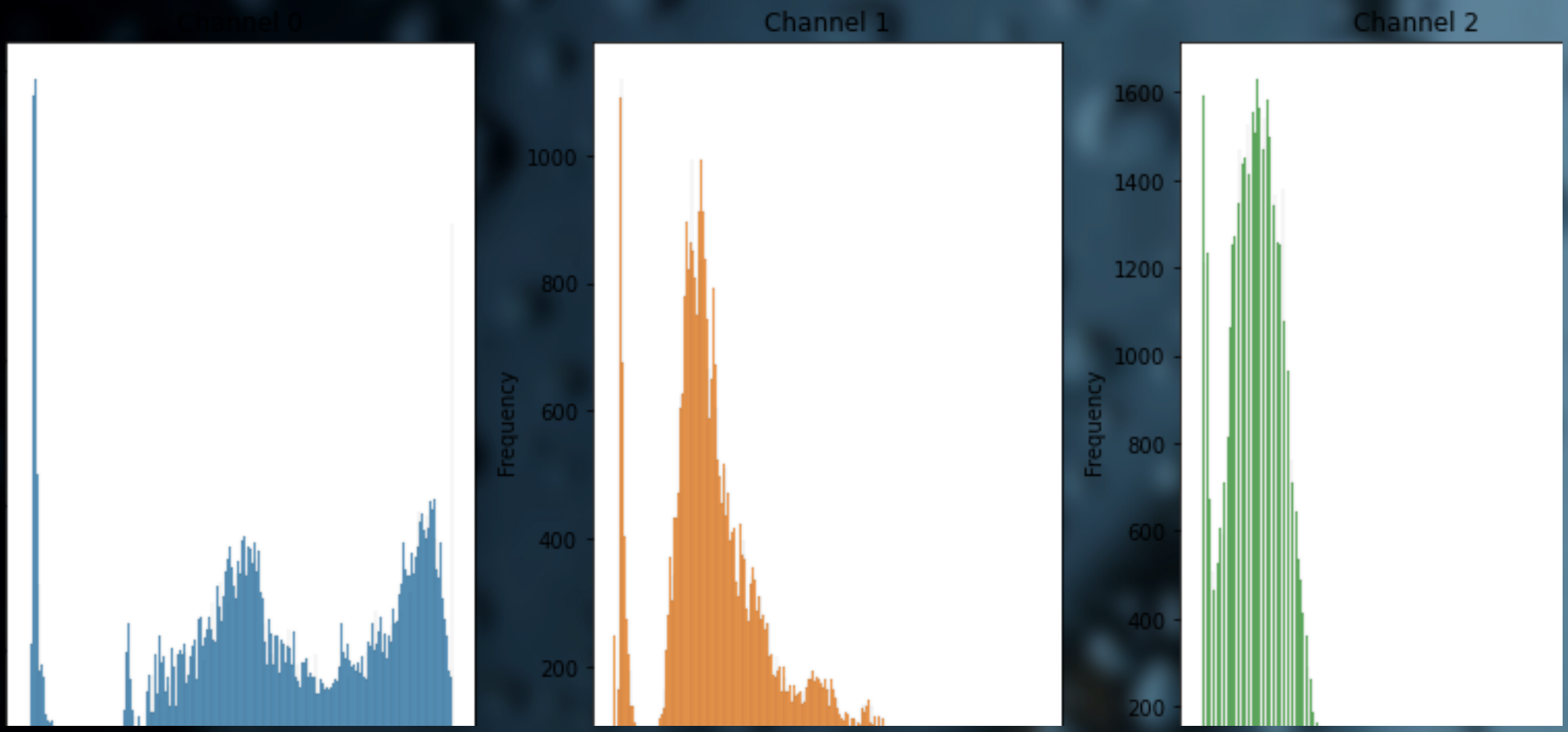
3. Image size distribution



Observation: The images have varying sizes (height and width).

EDA BEFORE DATA PRE-PROCESSING

4. Image Pixel distribution



Observation: the RGB color distribution

MODELLING

1, ANN Model

```
6/6 [==============================] - 3s 482ms/step - loss: 1.0370 - accuracy: 0.5089
Test Accuracy: 50.89%
Test Loss: 1.0370
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 150528)            0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               19267712
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 516
=================================================================
Total params: 19,268,228
Trainable params: 19,268,228
Non-trainable params: 0
_________________________________________________________________
```

Our ANN model had a test accuracy of 51%

MODELLING

1, 1ST CNN MODEL

```
6/6 [==============================] - 7s 1s/step - loss: 0.6283 - accuracy: 0.7515
Test Accuracy: 75.15%
Test Loss: 0.6283
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 256)       0
_________________________________________________________________
flatten_3 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               9437440
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 1028
```

Our 1st CNN model had a test accuracy of 75%

MODELLING

2, 2ND CNN MODEL

```
6/6 [==============================] - 3s 441ms/step - loss: 0.1118 - accuracy: 0.9645
Test Accuracy: 96.45%
Test Loss: 0.1118
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 1280)              2257984
_________________________________________________________________
dropout_2 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               327936
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 1028
=================================================================
Total params: 2,586,948
Trainable params: 2,552,836
Non-trainable params: 34,112
_________________________________________________________________
```

Our 2nd CNN model had a test accuracy of 96%

MODELLING

3. BEST CNN MODEL

```
4/4 [==============================] - 2s 563ms/step - loss: 1.5274 - accuracy: 0.9823 - precision: 0.9911 - recal
l: 0.9823
Test Loss: 1.5274
Test Accuracy: 0.9823
Test Precision: 0.9911
Test Recall: 0.9823
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b0 (Functional) (None, 1280)              4049564
_________________________________________________________________
batch_normalization_5 (Batch (None, 1280)              5120
_________________________________________________________________
dense_14 (Dense)             (None, 256)               327936
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_15 (Dense)             (None, 4)                 1028
=================================================================
Total params: 4,383,648
Trainable params: 4,339,072
Non-trainable params: 44,576
_________________________________________________________________
```

Our best CNN model had a test accuracy of 99%

CONCLUSIONS

Overall observation

Our multiclass weather image classification model using a CNN architecture achieved an overall accuracy of 99% on the test dataset. The model performed particularly well on "cloudy class" with a precision of 0.36, recall of 0.35, and F1-score of 0.36. However, it struggled with rain, achieving a lower precision of 0.18 and recall of 0.19.

STRENGTHS

- Effective feature extraction: the high accuracy shows the model's ability to extract key features like cloud patterns and textures from weather images.
- Robustness to noise: The model maintained good performance even when presented with noisy or low-resolution images.
- Transfer learning: Utilizing pre-trained weights from MobileNet and EfficientNet significantly improved training speed and accuracy.

LIMITATIONS

- Limited data availability: The model's performance could potentially improve with a larger and more diverse dataset, particularly for rain class.
- Overfitting: the history plots show some signs of overfitting on the training data, which could be mitigated by employing techniques like dropout or data augmentation.
- Limited interpretability: While the model achieves good accuracy, understanding its decision-making process for specific classifications remains a challenge.

RECOMMENDATIONS

- Data acquisition and augmentation: Collecting more data, especially for rain and shine classes, and explore data augmentation techniques to increase the diversity of the training set.
- Model architecture optimization: Experiment with different CNN architectures and hyperparameters to potentially improve performance and reduce overfitting.