# IMAGE CLASSIFIER REPORT

## LANDSCAPE FIELD WEATHER CLASSIFIER

Submitted By: Viren Suresh Patel

# Table of Contents

# 1. Executive Summary

The objective of this project was to develop a Convolutional Neural Network (CNN) model to classify landscape images into different categories based on the weather conditions on the days when those images were captured.

The need for accurate and timely weather prediction is critical for various industries, including agriculture and transportation. Traditional weather forecasting methods often have limitations in accuracy and efficiency, requiring complex methods and numerous independent variables.

Leveraging deep learning techniques such as CNNs offers the potential to improve the accuracy of weather prediction.

We designed a CNN architecture tailored for image classification tasks, consisting of convolutional layers for feature extraction followed by fully connected layers for classification. The model was trained using TensorFlow and Keras, optimizing it to minimize categorical cross-entropy loss.

Upon training and evaluation, the CNN model demonstrated promising performance in classifying landscape images based on weather conditions. The model achieved accuracy of 93% on the testing dataset, indicating its capability to effectively distinguish between different weather categories.

In conclusion, the developed CNN model presents a viable solution for automating weather classification tasks based on landscape images.

# 2. Model Building

## 2.1 Data Description

The weather landscape dataset consists of images captured under varying weather conditions, including Cloudy, Rainy, and Sunny. The dataset is split into two parts: Xtrain and Xtest, representing the training and testing datasets, respectively.
The Images are loaded as PIL instances, facilitating easy image processing.

Key details about the dataset are as follows:

- **Number of Data Points**
  - Training Dataset has 505 Images
  - Testing Dataset has 45 images

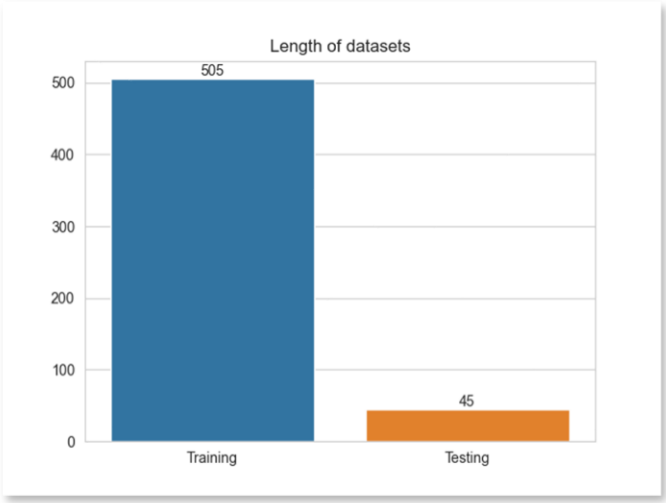



*Figure 2-1: Length of Training and Testing datasets*

- **3 Unique Classes**
  - Cloudy
  - Rainy
  - Sunny

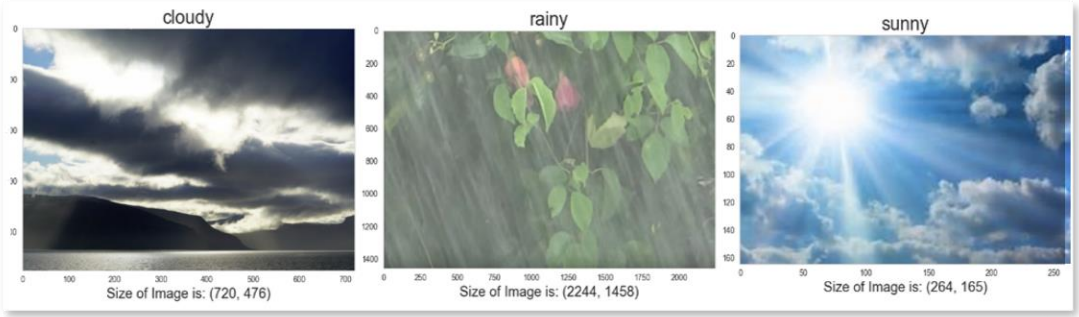



*Figure 2-2: Class wise images.*

- **Class Balance in Training Dataset**
  - Cloudy: 285 images
  - Rainy: 70 images
  - Sunny: 150 images

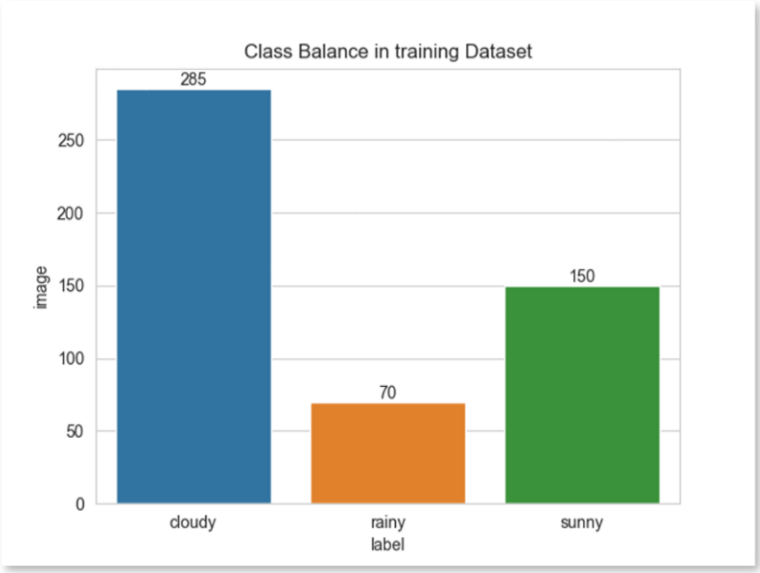



*Figure 2-3: Data Balance Between Classes*

- **Image Dimensions**
  - The image resolutions vary, ranging from 160x180 pixels to 2500x1400 pixels.

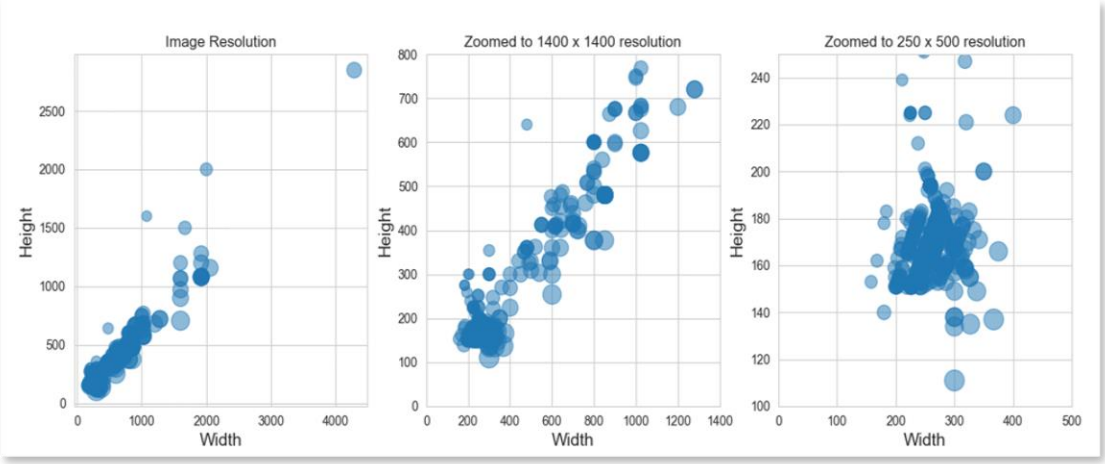



*Figure 2-4: Scatter Plot of Image Dimensions*

# 2.2 Data Preparation

By performing the data preprocessing and preparation steps, the raw image data was transformed into a format suitable for feeding into the Keras CNN Model.

The resized and normalized image arrays, along with their corresponding one-hot encoded labels, serve as the input and target data, respectively, for training and evaluating the CNN model for weather classification

- **Resizing Images:**
  - All images in both the training and testing datasets were resized to a common size of 160x160 pixels using an appropriate resizing method, ensuring uniformity in input dimensions for the CNN model.

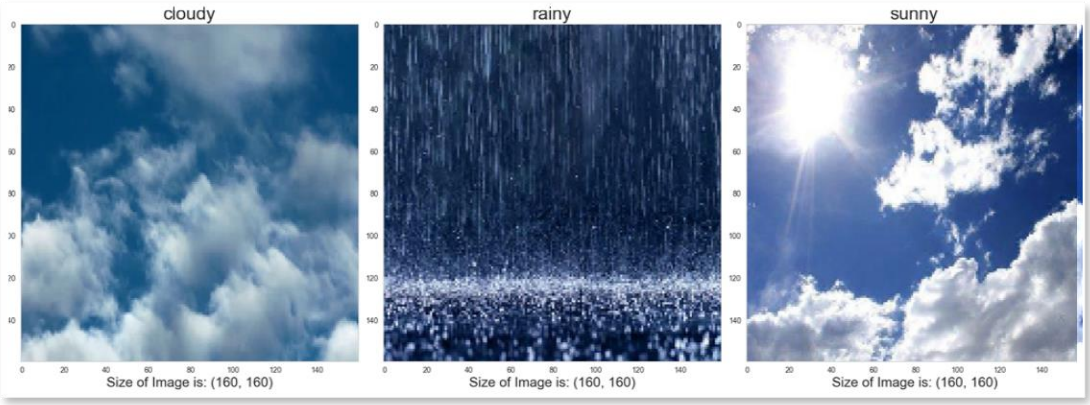
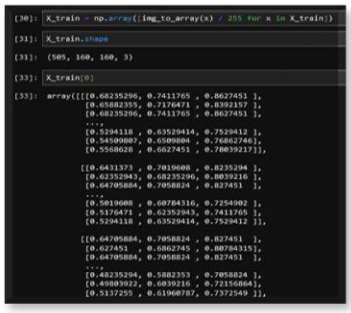



*Figure 2-5: Rescaled Images to 160x160px*

- **Converting Images to Arrays:**
  - Each resized image was converted into a numerical array representation using TensorFlow's `img_to_array()` function. This conversion enables the images to be processed as numerical data by the CNN model.
- **Rescaling Colour Scheme:**
  - The pixel values of the image arrays were rescaled from the original range of [0, 255] (representing colour intensity) to the normalized range of [0, 1]. This normalization ensures that the input data falls within a standardized range, facilitating stable and efficient training of the CNN model.



- **Encoding Labels with One-Hot Encoding:**

```
{'Cloudy': [1.0, 0.0, 0.0], 'Sunny': [0.0, 0.0, 1.0], 'Rainy': [0.0, 1.0, 0.0]}
```

# 2.3 Basic Model

Basic Model Architecture and Performance:

- **Architecture of Basic CNN Model:**
  - The basic CNN model was implemented using a sequence of:
    - Convolutional layer with 2 filters of size 3x3.
    - Max pooling layer with pool size of 2x2.
    - Convolutional Layer with 4 filters of size 3x3.
    - Max Pooling layer with a pool size of 2x2.
    - Flatten layer to convert 2D feature maps to a 1D vector.
    - Dense layer with 8 neurons.
  - The activation function used in convolution, and dense layer is relu.
  - The final Dense layer has activation of softmax.
- **Model Training Details:**
  - Learning rate was 0.01
  - Number of epochs: 10
  - Number of Iterations (Trials): 20
  - Validation Split: 0.2 (20% of Training data used for validation).
- **Performance**
  - The model's performance on the training data was reasonably good, achieving satisfactory accuracy.
  - However, the model's performance of the validation data was poor, indicating potential overfitting or lack of generalization.

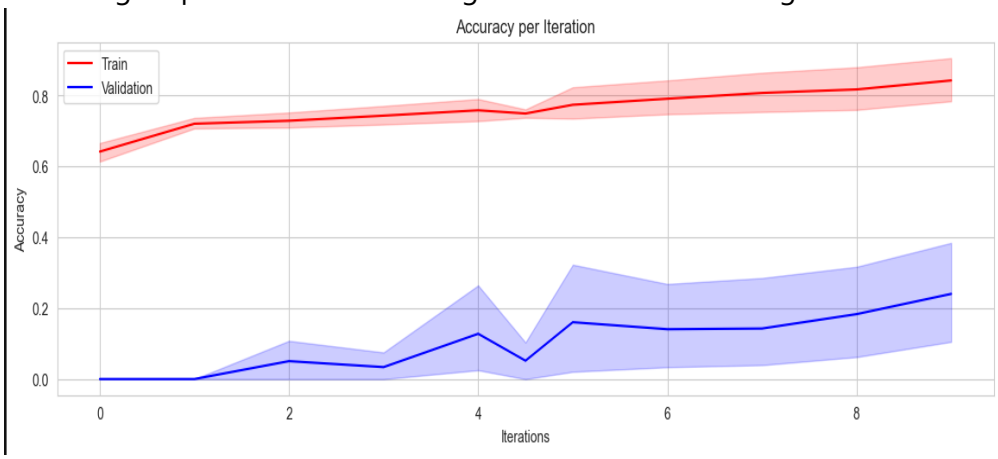



*Figure 2-6: Training and Validation Accuracy per Iteration.*

# 2.4 Challenges

- **Imbalance of data in classes in the dataset is contributing to poor performance.**
- **Training data is very less for generalizing.**

One way to resolve these issues is to perform data augmentation. This will increase the size of the training dataset so that the model will have enough data to train on. Using augmentation, we can also ensure that the classes in the training data are balanced.

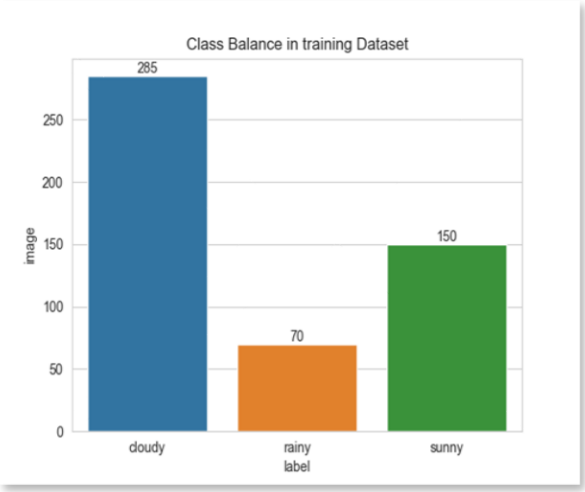



*Figure 2-7: Imbalance in training dataset*

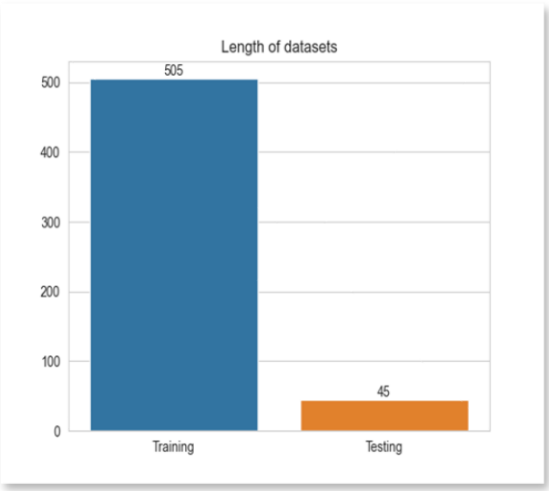



*Figure 2-8: Low data in training dataset*

# 3. Data Augmentation and Model Optimization

## 3.1 Data Augmentation.

Augmentation was implemented in the dataset to increase its diversity; the following process was implemented:

- **Get Class-wise Images from the Dataset:**
  - Class-wise images were extracted from the original dataset, segregating images belonging to each weather category: Cloudy, Rainy, and Sunny.

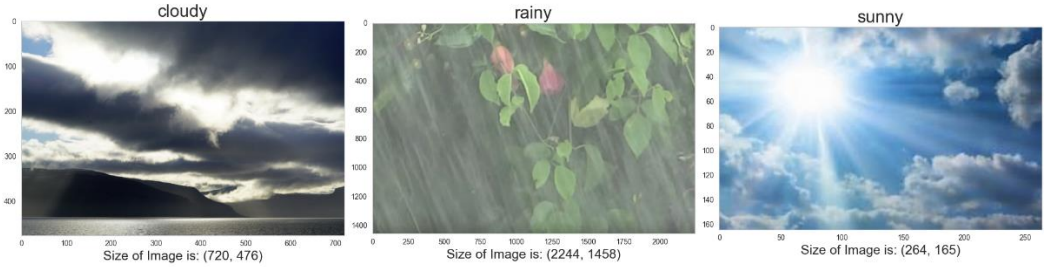



*Figure 3-1: Plotted class wise extracted images.*

- **Randomly Transform the Images:**
  - Random transformations were applied to each image to introduce variations and simulate different perspectives. These transformations included:
    - **Rotation by 90 degrees:** Images were rotated clockwise or counter-clockwise by 90 degrees to simulate different orientations.
    - **Horizontal Flip:** Images were flipped horizontally to create mirror images.
    - **Vertical Flip:** Images were flipped vertically to create inverted versions.

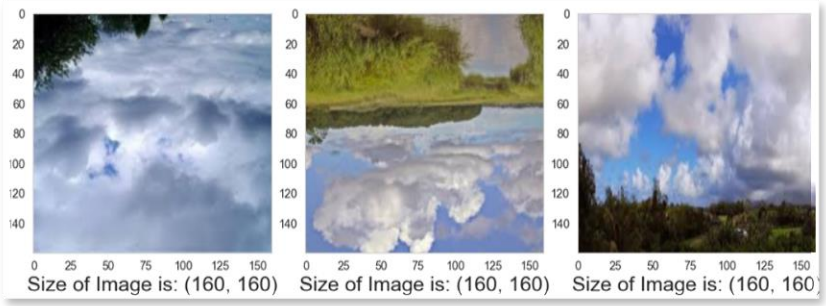



*Figure 3-2: Randomly Transformed Images*

- **Calculate Maximum Size and Multiple by Target Factor**
    - The maximum size among the class-wise image sets was determined, representing the maximum number of images among the three categories.
    - This maximum size was then multiplied by a target augmentation factor to generate additional augmented images.
- **Generate Augmented Images:**
    - Using the augmented factor and the calculated maximum size, additional images were generated for each class.
    - Random transformations, as described in step 2, were applied to the original images to create augmented versions.
    - The number of augmented images generated for each class was determined based on the target augmentation factor, ensuring a balanced distribution across classes.

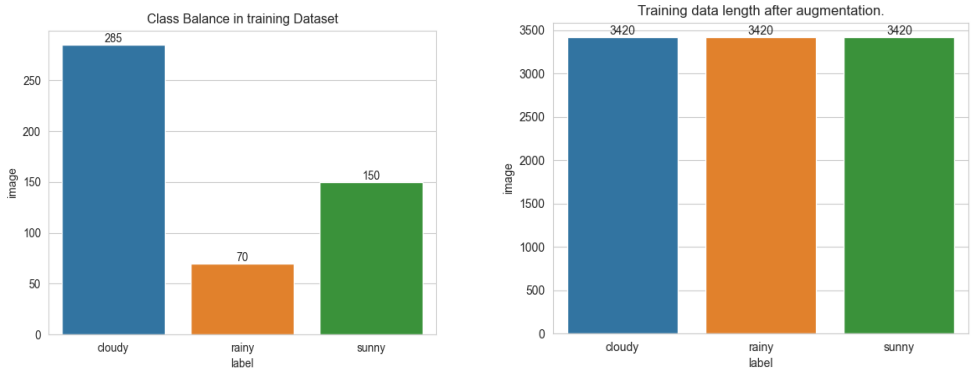



*Figure 3-3: Dataset Before and After Augmentation.*

# 3.2 Basic Model.

Performance of Basic Model on Augmented Dataset:

- **Model Architecture:**
    - The basic CNN model was configured with the following layer configurations: **['c_16_3', 'm_2', 'c_16_3', 'm_2', 'f', 'd_32'].**
    - **'c_16_3'**: Convolutional layer with 16 filters of size 3x3.
    - **'m_2'**: Max pooling layer with a pool size of 2x2.
    - **'f'**: Flatten layer to convert 2D feature maps to a 1D vector.
    - **'d_32'**: Dense layer with 32 neurons.
- **Training Parameters:**
    - Learning Rate: 0.001
    - Number of Epochs: 10
    - Number of Iterations (Trials): 10
    - Validation Split: 0.2 (20% of training data used for validation)
- **Performance Matrix:**
    - Loss: 0.0053 (Training), 0.0712 (Validation)
    - Accuracy: 0.9988 (Training), 0.9864 (Validation)

- **Observations:**
  - The model demonstrated excellent performance on the augmented dataset, achieving high accuracy and low loss values on both the training and validation data.
  - The training and validation metrics were almost identical, indicating that the model generalized well to unseen data and did not exhibit overfitting.
  - The high accuracy and low loss values suggest that the model effectively learned the underlying patterns and features in the augmented dataset, enabling accurate weather classification.

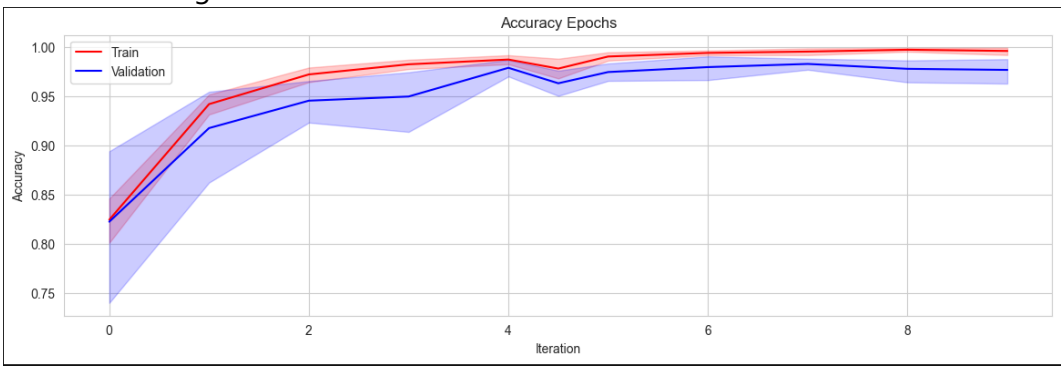



*Figure 3-4: Improved Accuracy of Validation data per Iterations.*

Overall, the basic CNN model trained on the augmented dataset performed exceptionally well, showcasing the effectiveness of data augmentation in enhancing model performance and generalization. The model's ability to accurately classify weather conditions based on landscape images can provide valuable insights and support various applications, including agriculture and transportation.

# 3.3 Hyperparameter Tuning

For hyperparameter tuning, we explored different combinations of model architectures and learning rates to optimize the performance of the CNN model. Here are the details of the hyperparameters tuned and the corresponding performance results:

| Model Architecture | Number of Trainable Parameters | Median Training Accuracy | Median Validation Accuracy |
|---|---|---|---|
| ['c_16_3', 'm_2', 'c_16_3', 'm_2', 'f', 'd_48'] | Total params: 1,231,763 | Train Accuracy: 0.9867 | Validation Accuracy: 0.9712 |
| ['c_24_3', 'm_4', 'c_16_3', 'm_2', 'f', 'd_36'] | Total params: 234,691 | Train Accuracy: 0.9926 | Validation Accuracy: 0.9786 |

**Learning Rates:** [0.01, 0.001, 0.002]

**The optimal configuration found was:**

| Model Architecture | Learning Rate | Training Accuracy | Validation Accuracy |
|---|---|---|---|
| ['c_24_3', 'm_4', 'c_16_3', 'm_2', 'f', 'd_36'] | 0.001 | 0.9926 | 0.9786 |

This configuration resulted in a model with 234,691 trainable parameters. The model achieved a median training accuracy of approximately 99.26% and a median validation accuracy of approximately 97.86%. These results demonstrate the effectiveness of the tuned hyperparameters in optimizing the model's performance on the augmented dataset, achieving high accuracy while maintaining good generalization to unseen validation data.

# 4. Final Model.

The final optimal model architecture consists of the following layers:

- Convolutional layer with 24 filters of size 3x3.
- Max pooling layer with a pool size of 4x4.
- Convolutional layer with 16 filters of size 3x3.
- Max pooling layer with a pool size of 2x2.
- Flatten layer to convert the output of the convolutional layers into a 1D array.
- Fully connected dense layer with 36 units.

The learning rate used to train the model was 0.001

The performance metrics of this model on the validation data are as follows:

| **Loss** | **Accuracy** | **Validation Loss** | **Validation Accuracy** |
|---|---|---|---|
| 0.0100 | 0.9979 | 0.0387 | 0.9898 |

**Finally, the accuracy of the model on the testing data is 0.93**.

# 5. Way Forward

Implementing the CNN, gained the skill of optimizing the CNN for more accuracy using optimizers, iterations and Augmentations.

Only problem is the rainy class prediction can be improved as features of cloudy and rainy are almost same. It's essential to address the issue of overlapping features with the "cloudy" class.

Here are a few potential strategies to consider for further improvement.

- **Transfer Learning**
    - Utilize pre-trained models or features learned from models trained on larger datasets. Fine-tuning a pre-trained model on your dataset, especially one trained on a diverse set of weather conditions, may help improve the model's ability to distinguish between classes.