Weather Classification

Team Members

Maram Khaled | 20210890

Mohammed Hamed | 20210758

Mohammed Amin | 20210748

Mohammed Ahmed Salem | 20210730

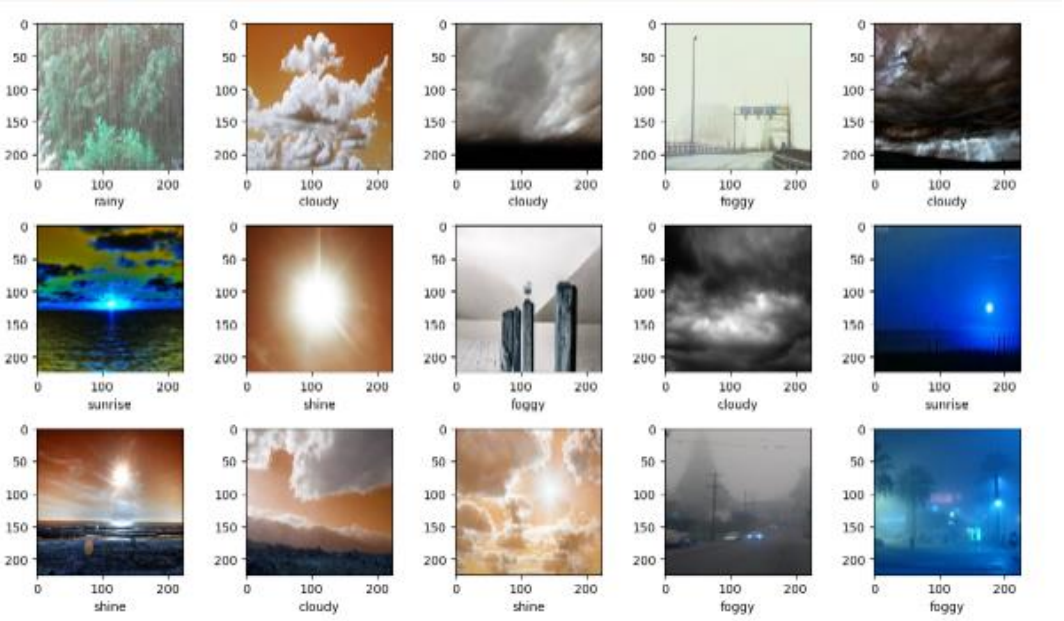
Mona Mohammed | 20210967

Mohammed Ahmed Mohammed | 20210738

Mahmoud Nasser | 20210882

Project Overview

- ## Task:
    - Weather Condition Classification
- ## Dataset:
    - Multiclass Weather Dataset (Kaggle) | (Weather Classification | Kaggle)
- ## Input:
    - Weather Images



- ## Output Classes:
    - Cloudy, Foggy, Rainy, Shine, Sunrise

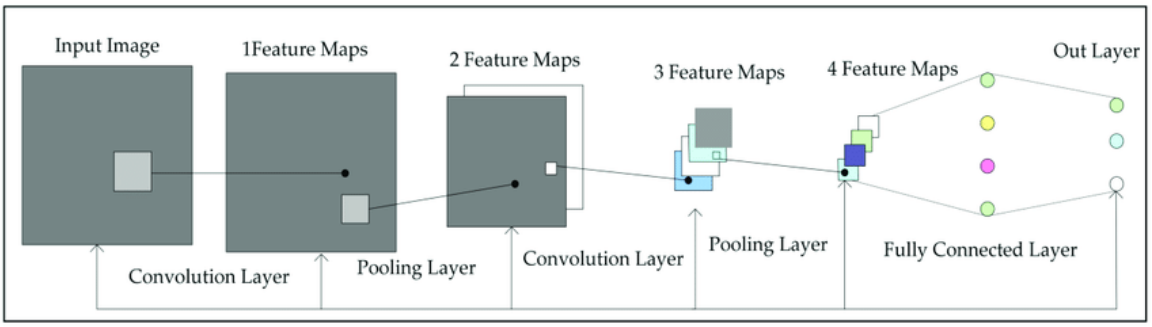
Methodology

- ## Data Preprocessing

1. Image Resize:
    a. Resized all images to 224x224 pixels.
2. Normalization:
    a. Pixel values scaled to the range [0, 1].
3. Split dataset:
    a. 70% training, 15% validation, 15% testing.
4. Data Augmentation:
    a. **Augmentation** (applied to training set):
        i. Horizontal flipping.
        ii. Rotation.
        iii. Brightness adjustment.

DenseNet121 Model

- ***Introduction to DenseNet***
    - *DenseNet-121 is a Densely Connected Convolutional Network designed to overcome the vanishing gradient problem common in deep networks. Unlike traditional CNNs, DenseNet connects each layer to every other layer in a feed-forward manner, enabling efficient feature reuse, reduced redundancy, and fewer parameters.*



- ***DenseNet Architecture & Components***
    - *Components of DenseNet include:*
    - *Connectivity*
    - *DenseBlocks*
    - *Growth Rate*
    - *Bottleneck layers*

DenseNet-121 Architecture (Details)

- ***each transition layer has a 1x1 convolutional layer and a 2x2 average pooling layer with a stride of 2.***

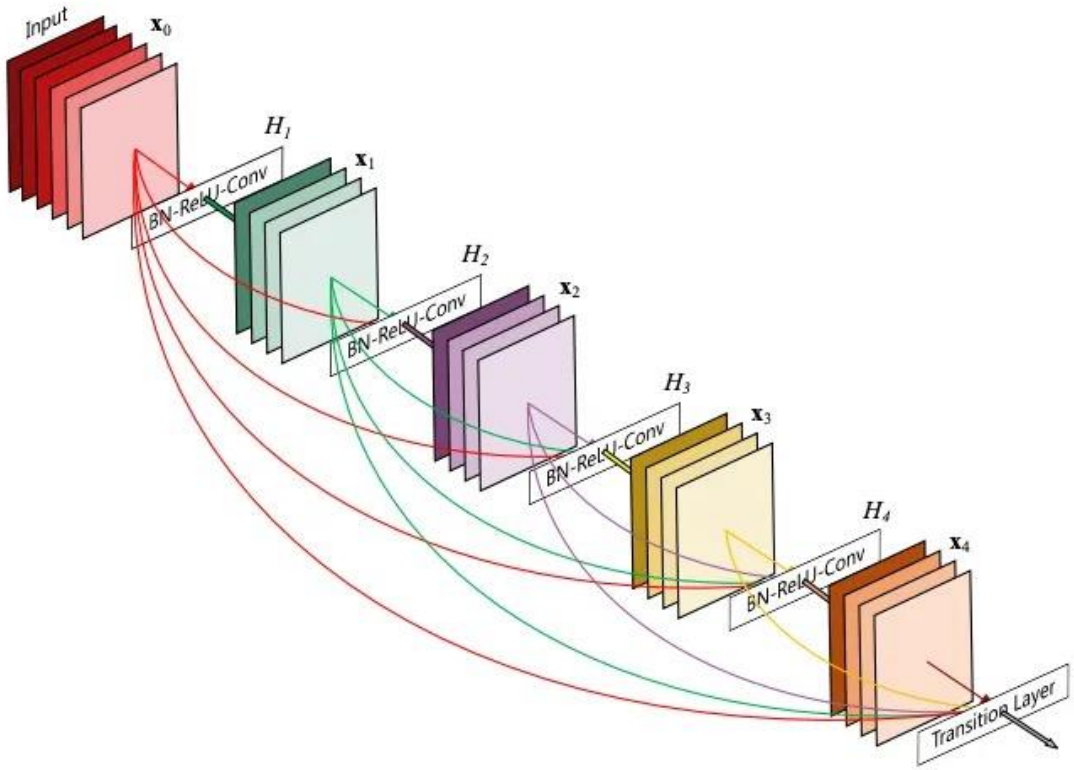
- ***Thus, the layers present are as follows:***
    - *Basic convolution layer with 64 filters of size 7X7 and a stride of 2*
    - *Basic pooling layer with 3x3 max pooling and a stride of 2*
    - *Dense Block 1 with 2 convolutions repeated 6 times*
    - *Transition layer 1 (1 Conv + 1 AvgPool)*
    - *Dense Block 2 with 2 convolutions repeated 12 times*
    - *Transition layer 2 (1 Conv + 1 AvgPool)*
    - *Dense Block 3 with 2 convolutions repeated 24 times*
    - *Transition layer 3 (1 Conv + 1 AvgPool)*
    - *Dense Block 4 with 2 convolutions repeated 16 times*
    - *Global Average Pooling layer- accepts all the feature maps of the network to perform classification*
    - *Output layer*

- ***Therefore, DenseNet-121 has the following layers:***
    - *1 | 7x7 Convolution*
    - *58 | 3x3 Convolution*
    - *61 | 1x1 Convolution*
    - *4 AvgPool*
    - *1 Fully Connected Layer*
    - *In short, DenseNet-121 has 120 Convolutions and 4 AvgPool.*

- ***All layers i.e. those within the same dense block and transition layers, spread their weights over multiple inputs which allows deeper layers to use features extracted early on.***

DenseNet-121 Architecture

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			



- ***References***

i. G. Huang, Z. Liu, van, and Weinberger, Kilian Q, "Densely Connected Convolutional Networks," arXiv.org, 2016. Arxiv.org/abs/1608.06993.

ii. Architecture of DenseNet-121

iii. DenseNet. Summary | by NOCODING AI | NOCODING AI | Medium

- ***Papers Reference:***

  - https://arxiv.org/abs/1608.06993 ([1608.06993] Densely Connected Convolutional Networks)
  - The DenseNet-121 architecture | Download Scientific Diagram
  - Deep Architecture based on DenseNet-121 Model for Weather Image Recognition

Xception Model

- ***Introduction to DenseNet***
    - Xception, an abbreviation for "***Extreme Inception***," represents a milestone in convolutional neural network (CNN) design. Conceived by François Chollet, the creator of the Keras deep learning library, Xception was introduced in 2017 as an evolution of the Inception architecture. In this blog post, we will delve into the architecture and approaches that distinguish Xception in the realm of deep learning.

- ***Architecture Overview***
    - Xception is based on depthwise separable convolutions, which allow for efficient computation and better handling of feature maps in comparison to traditional convolutional networks.

- ***Components of Xception:***

1. **Depthwise Separable Convolutions**:
    a. **Depthwise Convolution**: Applies a single filter across each input channel. This step allows the model to capture spatial patterns in individual channels.
    b. **Pointwise Convolution**: After applying depthwise convolution, a pointwise convolution (1x1 convolution) is applied which reduces the depth of the output channels. This step is crucial for reducing the number of parameters and improving computational efficiency.
2. **Blocks**:
    a. **Entry Flow**: Contains three blocks:
        i. **Block 1**: Starts with a depthwise convolution followed by pointwise convolution, followed by a max pooling layer.
        ii. **Block 2** and **Block 3**: Each block consists of multiple separable convolution layers followed by a max pooling layer.
    b. **Middle Flow**: Comprises residual connections to preserve input features, with multiple depthwise separable convolution layers to refine the feature maps.

c. **Exit Flow**: Includes the same structure as the middle flow but reduces the number of feature maps gradually through convolution layers and a global average pooling layer.

3. **Transition Layers**:
   a. The transition layers between blocks reduce the size of feature maps using 1x1 convolutions and max pooling layers, which helps in progressively capturing more abstract features.
4. **Global Average Pooling**:
   a. After the exit flow, the network applies a global average pooling layer to aggregate features across all channels. This reduces the dimensionality before feeding into the fully connected layer.
5. **Fully Connected Layer**:
   a. The final layer of Xception performs classification or regression based on the output from the global average pooling layer.

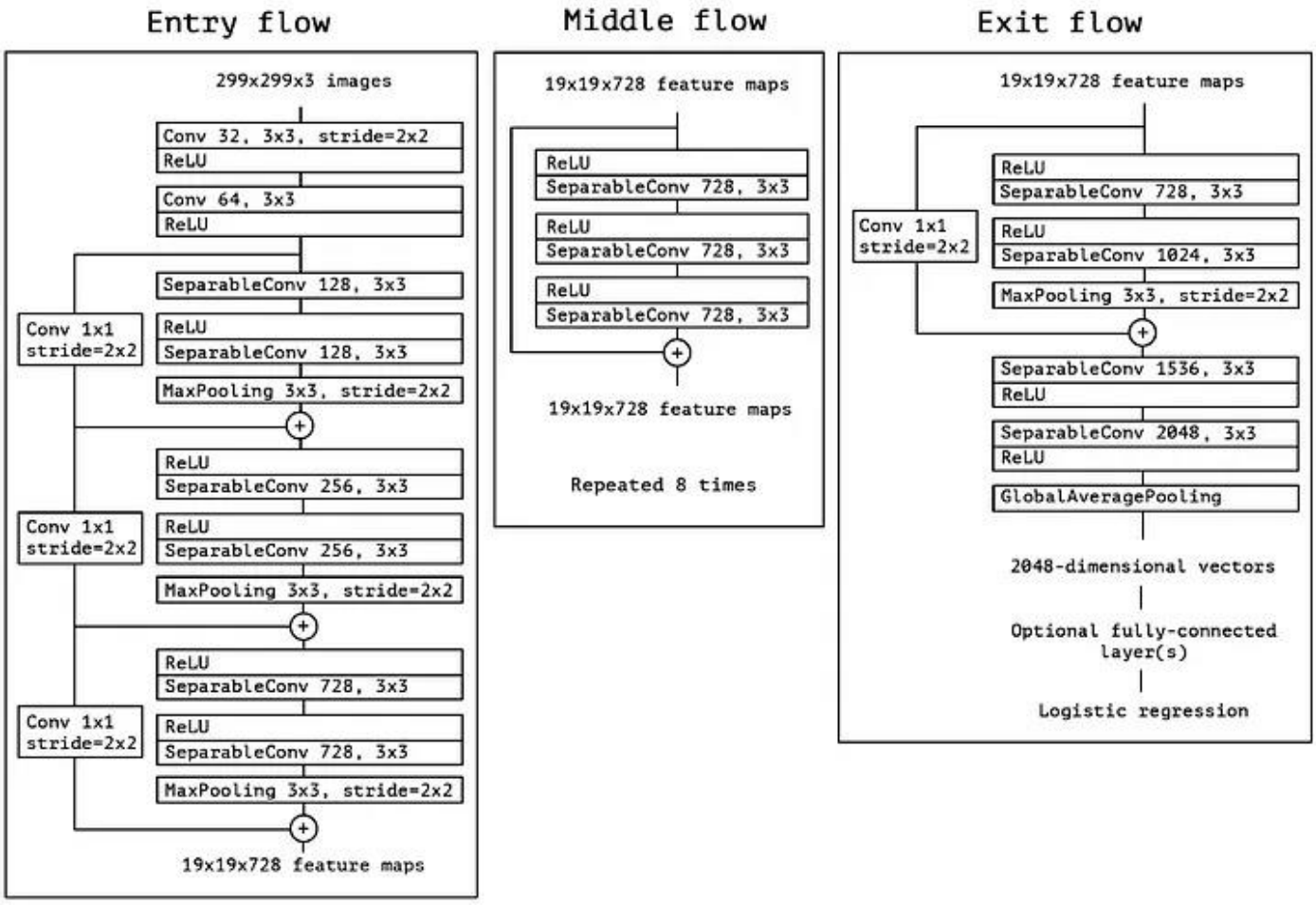
- ***Xception Model Summary:***

  - Xception consists of:
    - 71 layers in total.
    - **Initial Layers**:
      - 3x3 Convolution with 32 filters.
      - 3 blocks in the entry flow (each with depthwise and pointwise convolutions).
      - 8 blocks in the middle flow.
      - 3 blocks in the exit flow.
    - **Final Layers**:
      - Global average pooling layer.
      - Fully connected layer for classification.

- ***Key Characteristics:***

  - **Efficiency**: By utilizing depthwise separable convolutions, Xception reduces the number of parameters, making it computationally efficient.
  - **Feature Reusability**: The model maintains feature reusability across multiple layers, which helps it generalize well to unseen data.

Xception Model Architecture



- ***References:***

1. Chollet, Francois. "Xception: Deep Learning with Depthwise Separable Convolutions." **arXiv:1610.02357** [cs.CV], 2016. Available at: https://arxiv.org/abs/1610.02357.
2. Zhang, Hang, et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." **arXiv:1602.07261** [cs.CV], 2016. Available at: https://arxiv.org/abs/1602.07261.
3. Howard, Andrew G., and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification." **arXiv:1801.06146** [cs.CL], 2018. Available at: https://arxiv.org/abs/1801.06146.

Resnet Model

- ## Introduction to ResNet

  - ResNet, short for Residual Networks, represents a breakthrough in the design of deep convolutional neural networks (CNNs). Introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015, ResNet has revolutionized deep learning by addressing the vanishing gradient problem and enabling the training of much deeper networks.

- ## *Architecture Overview*

  - ResNet is built around the concept of residual learning, where each layer receives input from one or more preceding layers and adds a residual connection to maintain the input signal. This architecture helps in training very deep networks without sacrificing performance or accuracy.

- ## *Components of ResNet:*

1. **Residual Blocks**:
   a. **Basic Idea**: A residual block allows a shortcut connection to bypass one or more layers. This skip connection helps mitigate the vanishing gradient problem and allows deeper networks to be trained effectively.
   b. **Structure**: Each residual block consists of two or more convolutional layers with a skip connection that adds the input directly to the output of the last layer in the block. This residual connection facilitates a deeper representation without a significant increase in computation.
   c. **Layers**: The common form for a residual block is Conv-ReLU-Conv, where the first convolution reduces the dimensionality of the input and the second convolution reconstructs the same dimensions as the input.
2. **Bottleneck Layers**:
   a. To further reduce the number of parameters, ResNet introduces bottleneck layers which consist of three convolutions:
      i. **1x1 Convolution**: Reduces the number of feature maps.
      ii. **3x3 Convolution**: Extracts local features.

iii. **1x1 Convolution**: Reconstructs the feature maps to match the original dimensions.

3. **Transition Layers**:
    a. Transition layers between residual blocks typically involve downsampling with a 2x2 average pooling layer or a 1x1 convolution followed by average pooling. This step reduces the spatial dimensions of feature maps while retaining essential features.
4. **Global Average Pooling (GAP)**:
    a. At the end of the network, a Global Average Pooling layer is used to aggregate the feature maps across spatial dimensions. This pooling reduces the dimensionality of the network's output, making it suitable for classification tasks.
5. **Fully Connected Layer**:
    a. The final layer of ResNet performs classification (or regression) based on the aggregated features from the GAP layer.

- ***ResNet Model Summary:***

    - ResNet can be summarized by its depth and structure:
        - **Depth**: ResNet architectures come in various depths, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, and ResNet-200. The numbers refer to the number of layers in the network.
        - **Basic Structure**:
            - Each block adds layers while maintaining a residual connection.
            - **ResNet-50**: Contains 50 layers, with 16 residual blocks and bottleneck layers.
            - **ResNet-101**: Contains 101 layers, with 33 residual blocks and bottleneck layers.
            - **ResNet-152**: Contains 152 layers, with 45 residual blocks and bottleneck layers.
            - **ResNet-200**: Contains 200 layers, with 63 residual blocks and bottleneck layers.

- ***Key Characteristics:***

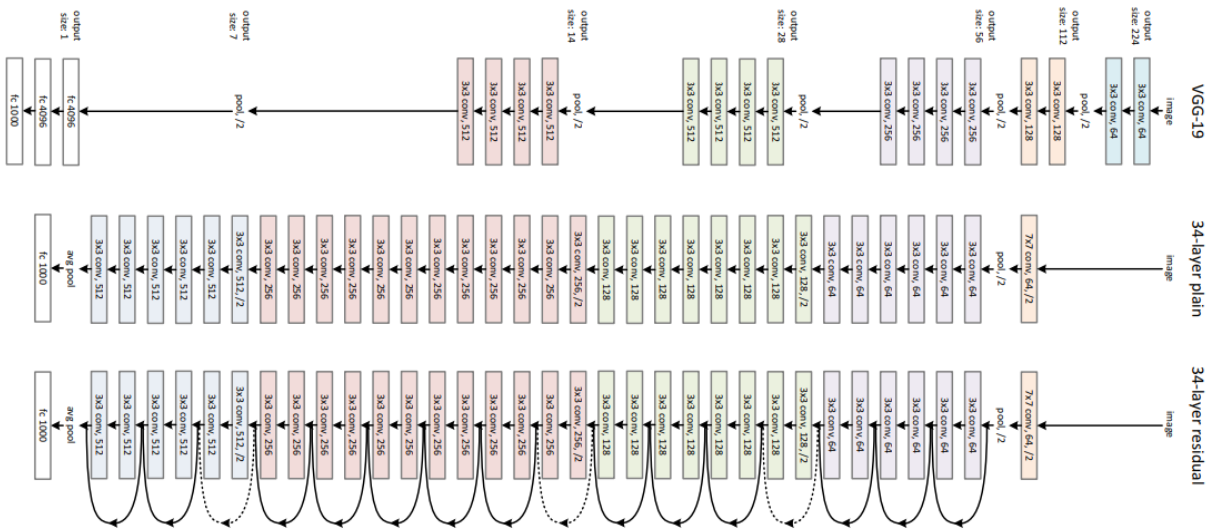
    - **Vanishing Gradient Problem**: ResNet addresses this issue by adding direct connections (skip connections) between layers.

- **Performance**: Empirically, ResNet architectures have shown superior performance in various image recognition tasks due to their ability to learn complex representations efficiently.
- **Modular Design**: The modular nature of ResNet allows easy adaptation to different depths and sizes, making it highly versatile for various applications.

- ***References:***

1. He, Kaiming, et al. "Deep Residual Learning for Image Recognition." **arXiv:1512.03385** [cs.CV], 2015. Available at: https://arxiv.org/abs/1512.03385.
2. Zhang, Xiangyu, et al. "ResNeXt: Design Space Exploration of Deep Convolutional Neural Networks for Classification." **arXiv:1611.05431** [cs.CV], 2016. Available at: https://arxiv.org/abs/1611.05431.
3. Szegedy, Christian, et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." **arXiv:1602.07261** [cs.CV], 2016. Available at: https://arxiv.org/abs/1602.07261.

ResNet Model Architecture



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Comparison Between ResNet, Xception, and DenseNet Models

1. ResNet Model

- **Architecture Overview**:
  - **Residual Connections**: ResNet uses residual connections, allowing for the direct flow of data across layers. This "skip connection" helps mitigate the vanishing gradient problem and allows deeper networks to be trained efficiently.
  - **Depth**: Variants include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, each increasing in depth and complexity.
  - **Bottleneck Layers**: Introduces bottleneck layers to reduce the number of parameters and computational cost.
  - **Global Average Pooling**: Final layer uses global average pooling followed by a fully connected layer, making it suitable for image classification tasks.
- **Pros**:
  - **Efficiency**: ResNet's use of skip connections and bottleneck layers allows for deeper models without a significant increase in computational cost.
  - **Performance**: Empirical evidence shows that deeper ResNet architectures perform well across a variety of tasks, including image classification and object detection.
  - **Versatility**: The architecture's modular design allows it to be adapted easily for different tasks and datasets.
- **Cons**:
  - **Complexity**: The deeper versions of ResNet can be more computationally intensive and require more memory.
  - **Overfitting**: If not properly regularized, deeper ResNet models can suffer from overfitting due to the larger number of parameters.

2. Xception Model

- **Architecture Overview**:
  - **Depthwise Separable Convolutions**: Xception uses depthwise separable convolutions, which consist of depthwise and pointwise convolutions. Depthwise convolution applies a single filter per channel, and pointwise convolution reduces the depth of feature maps, significantly reducing the number of parameters.
  - **Flow Structure**: Divided into three parts: Entry flow, Middle flow, and Exit flow. Each section refines features at different levels of abstraction.
  - **Global Average Pooling**: Final layer uses global average pooling to aggregate features before feeding into the fully connected layer.
- **Pros**:
  - **Efficiency**: By using depthwise separable convolutions, Xception significantly reduces the number of parameters compared to traditional CNNs, making it computationally efficient.
  - **Generalization**: The model shows excellent generalization to unseen data due to the reusability of features across layers.
  - **Performance**: Xception achieves high performance across various benchmarks, particularly in image recognition tasks.
- **Cons**:
  - **Complexity**: Despite its computational efficiency, the architecture might be more complex to tune than simpler CNN architectures.
  - **Overhead**: The middle flow section can increase latency if not optimized well.

3. DenseNet Model

- **Architecture Overview**:
  - **Dense Blocks**: Each layer in DenseNet is connected directly to every other layer, allowing information to be propagated more effectively across layers.
  - **Growth Rate**: Specifies the number of feature maps each layer adds. This controlled growth helps in keeping the model efficient and allows for deep networks without excessive memory consumption.
  - **Global Average Pooling**: The final layer applies global average pooling before feeding into the fully connected layer.
- **Pros**:
  - **Information Propagation**: Dense connections ensure that features are reused effectively across layers, leading to efficient learning and improved accuracy.
  - **Reduced Overfitting**: DenseNet's dense connections provide strong regularization, which reduces overfitting, especially useful when training on limited data.
  - **Performance**: DenseNet achieves competitive results in various image recognition benchmarks.
- **Cons**:
  - **Parameter Count**: The dense connections increase the total number of parameters compared to ResNet, which could be a limitation if computational resources are constrained.
  - **Computational Cost**: The dense connections make DenseNet more computationally intensive compared to some other architectures.

Comparison on Performance:

1. **Performance on Image Classification Tasks**:
    a. **ResNet**: Known for scalability and achieving high accuracy with deep networks. Suitable for large datasets like ImageNet.
    b. **Xception**: Efficient on large-scale image datasets due to reduced parameter count, performs well in terms of speed and accuracy.
    c. **DenseNet**: Best suited for situations requiring efficient learning from feature-rich datasets, effective in smaller datasets due to its dense connections that improve regularization.
2. **Data Requirements**:
    a. **ResNet**: Effective with large datasets due to its deep structure.
    b. **Xception**: Can perform well with smaller datasets due to its parameter efficiency.
    c. **DenseNet**: Beneficial with medium to large datasets as its dense connections help in efficient learning even with fewer examples.
3. **Task Suitability**:
    a. **ResNet**: Ideal for complex tasks that require deep feature extraction, such as object detection and segmentation.
    b. **Xception**: Suitable for tasks requiring computational efficiency, like image recognition and classification in real-time scenarios.
    c. **DenseNet**: Best for tasks where feature reusability and regularization are key, such as in medical imaging or when handling noisy data.

Performance Comparison

Accuracy Metrics

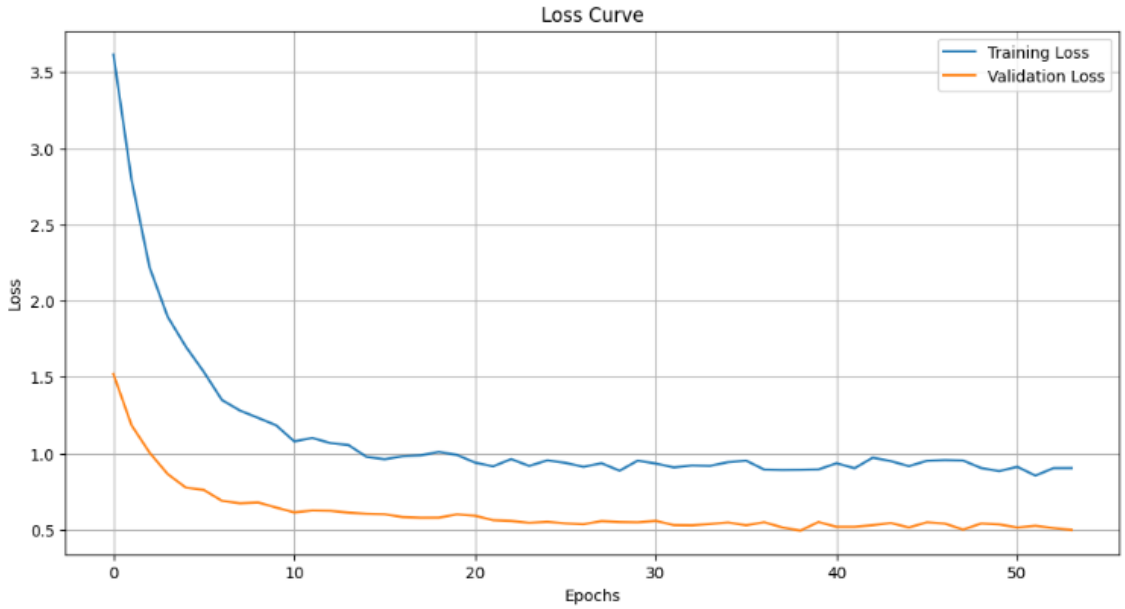
Model	Training Accuracy	Validation Accuracy	Test Accuracy
DenseNet121	90.8%	88.8%	89.3%
Xception	92.1%	84.4%	86%
ResNet50	92.5%	89.9%	90%

Detailed Performance Metrics

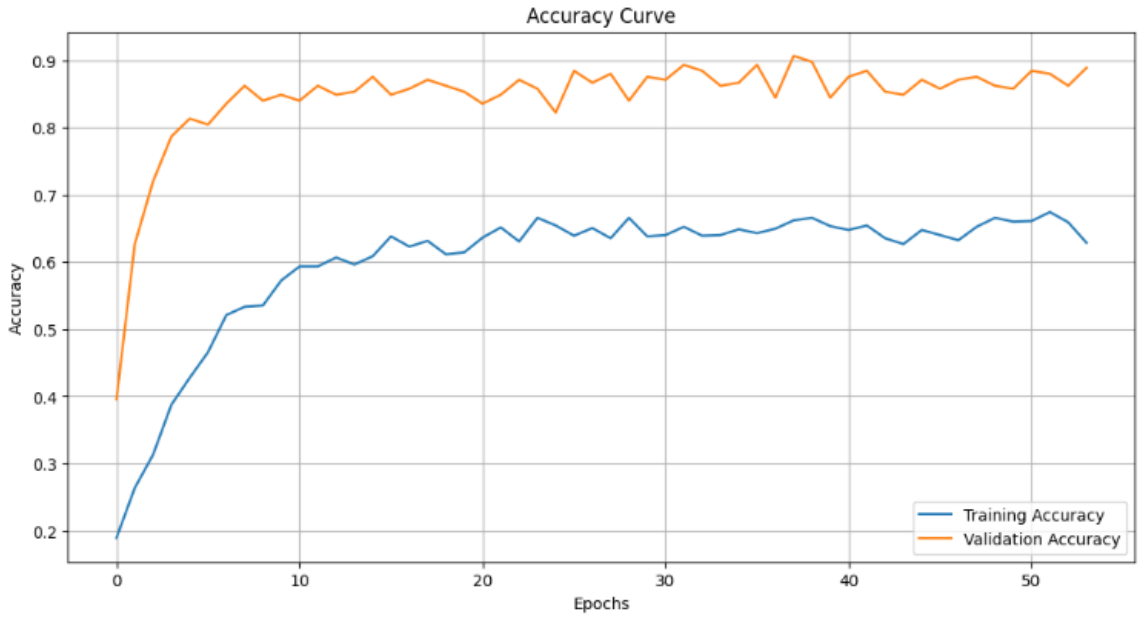
Model	Precision	Recall	F1-Score	AUC Score
DenseNet121	0.90	0.89	0.89	0.96
Xception	0.86	0.86	0.86	0.92
ResNet50	0.9	0.89	0.89	0.98

Performance as Visualization For (Densenet)

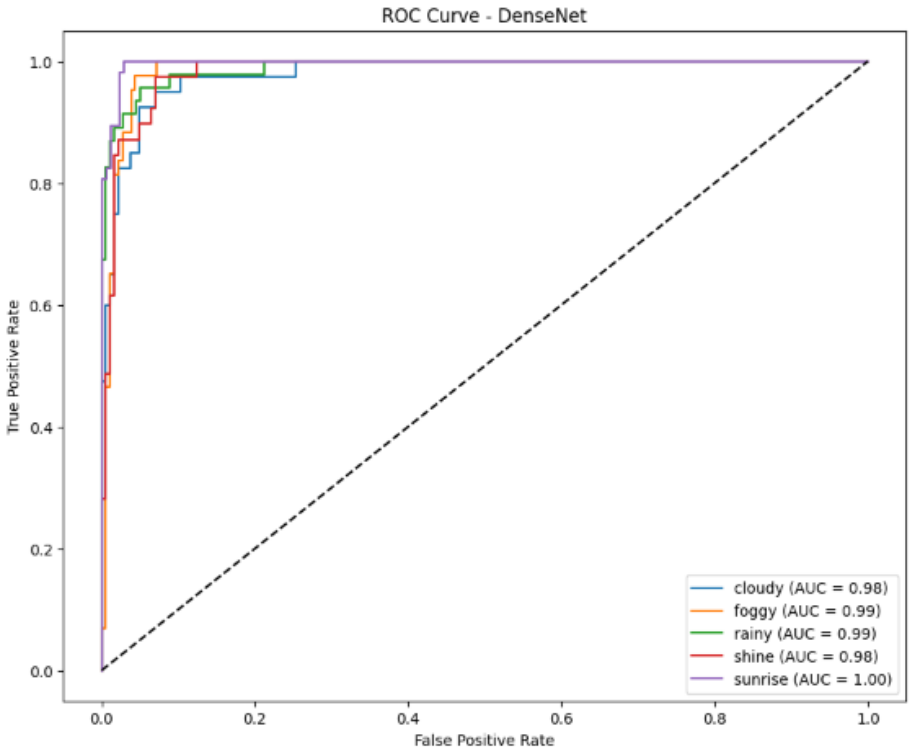
Loss Curve



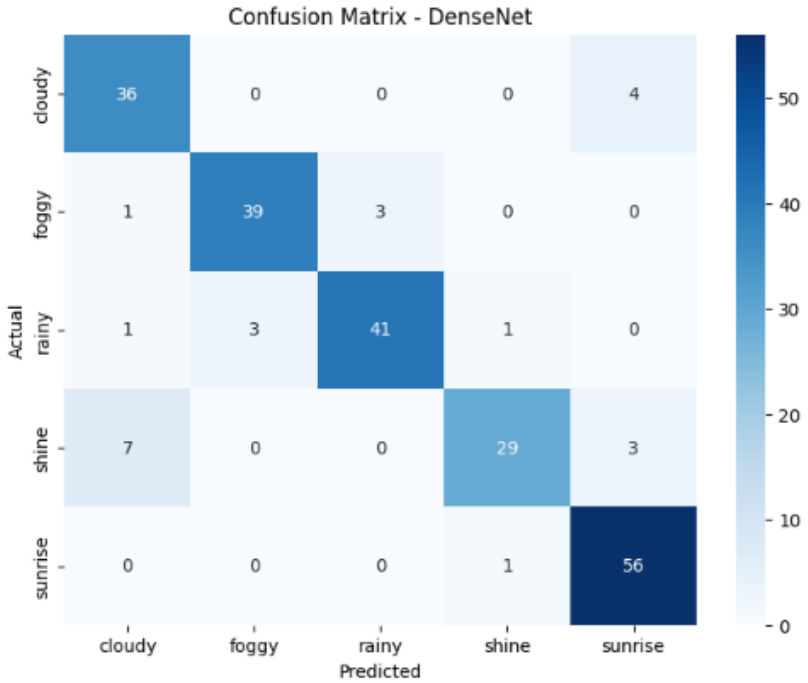
Accuracy Curve



ROC Curve



Confusion Matrix



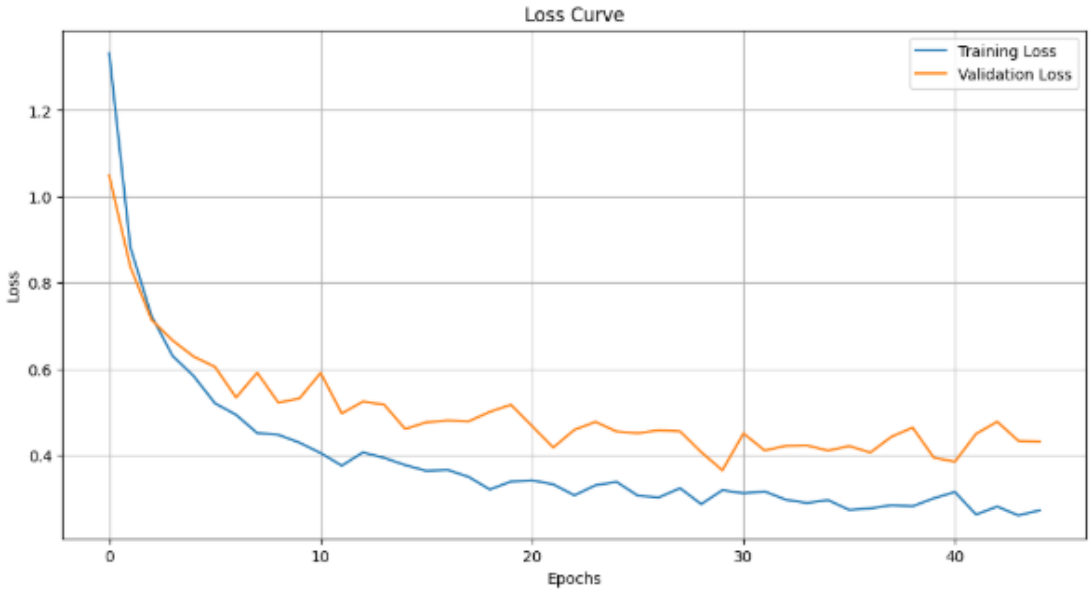
Classification Report

DenseNet Classification Report

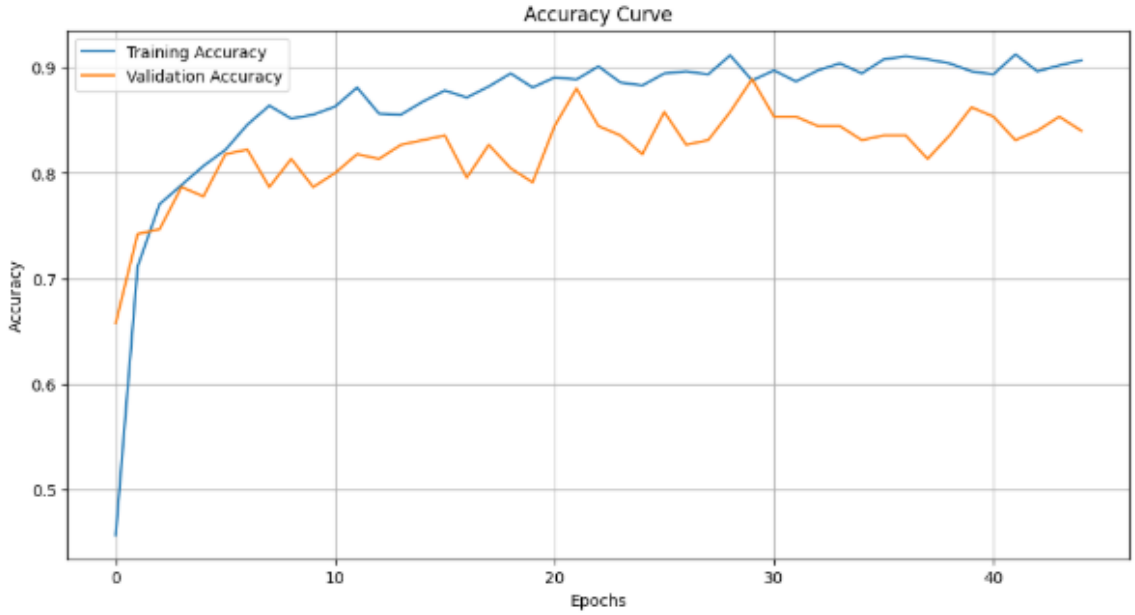
	precision	recall	f1-score	support
cloudy	0.80	0.90	0.85	40
foggy	0.93	0.91	0.92	43
rainy	0.93	0.89	0.91	46
shine	0.94	0.74	0.83	39
sunrise	0.89	0.98	0.93	57
accuracy			0.89	225
macro avg	0.90	0.88	0.89	225
weighted avg	0.90	0.89	0.89	225

Performance as Visualization For (Xception)

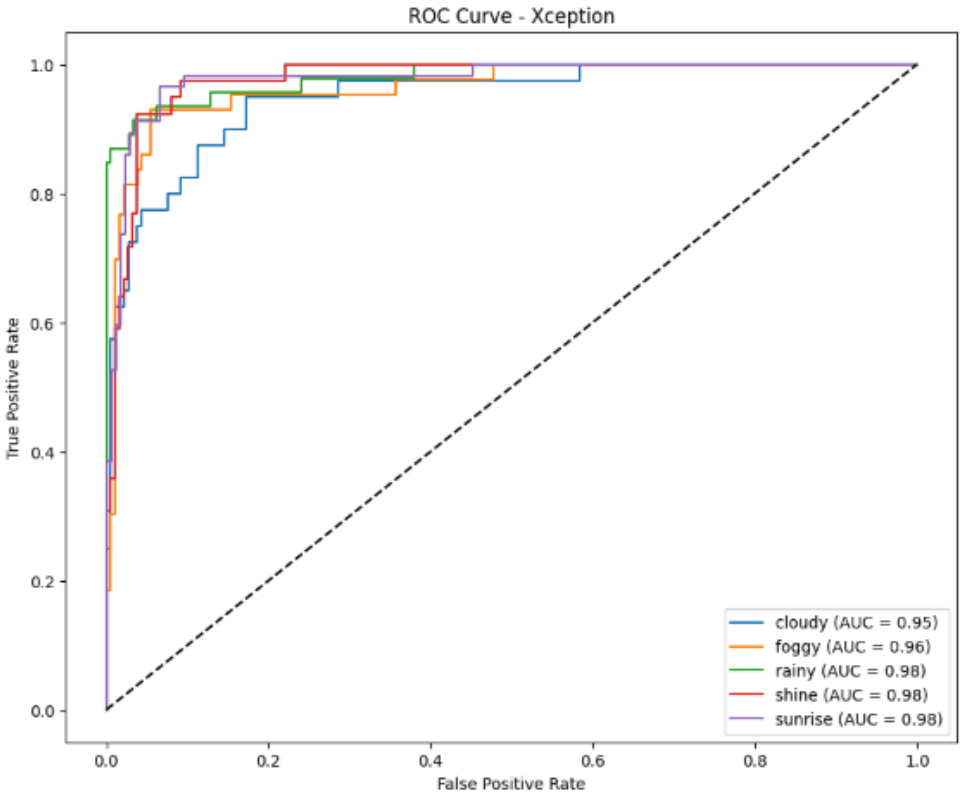
Loss Curve



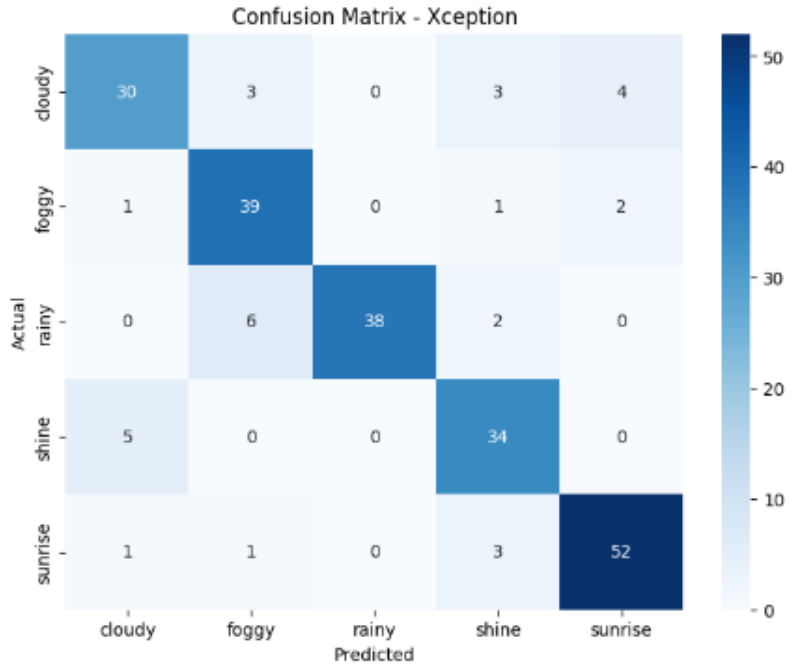
Accuracy Curve



ROC Curve



Confusion Matrix



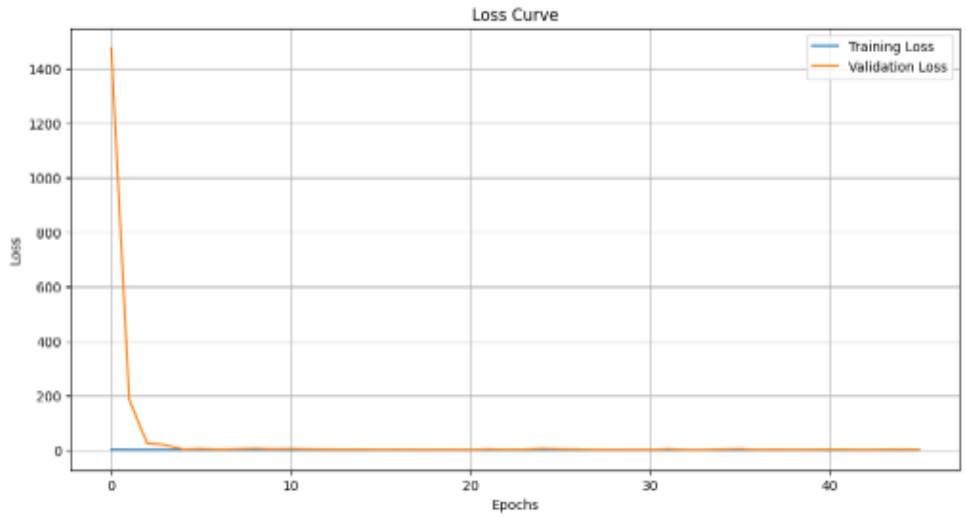
Classification Report

Xception Classification Report

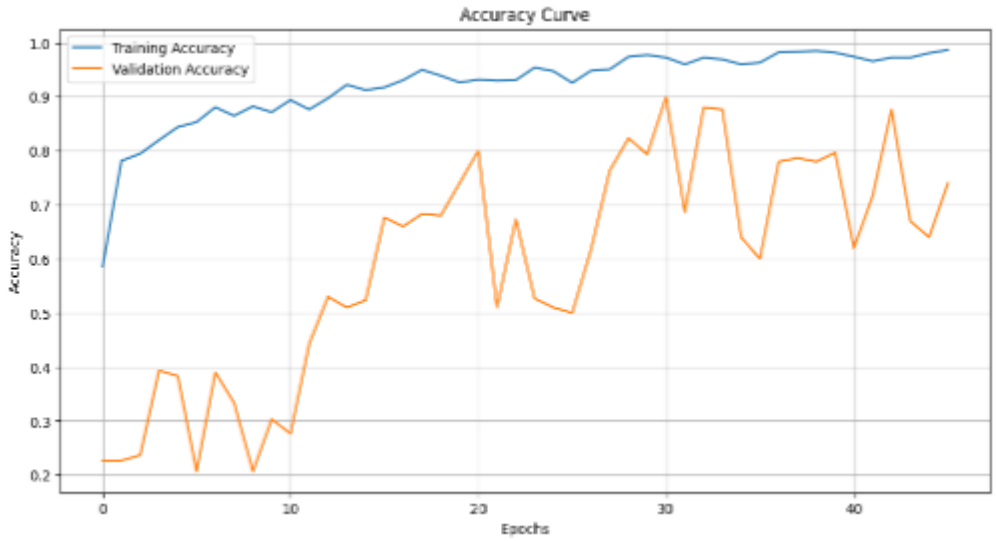
	precision	recall	f1-score	support
cloudy	0.81	0.75	0.78	40
foggy	0.80	0.91	0.85	43
rainy	1.00	0.83	0.90	46
shine	0.79	0.87	0.83	39
sunrise	0.90	0.91	0.90	57
accuracy			0.86	225
macro avg	0.86	0.85	0.85	225
weighted avg	0.86	0.86	0.86	225

Performance as Visualization For (Resnet)

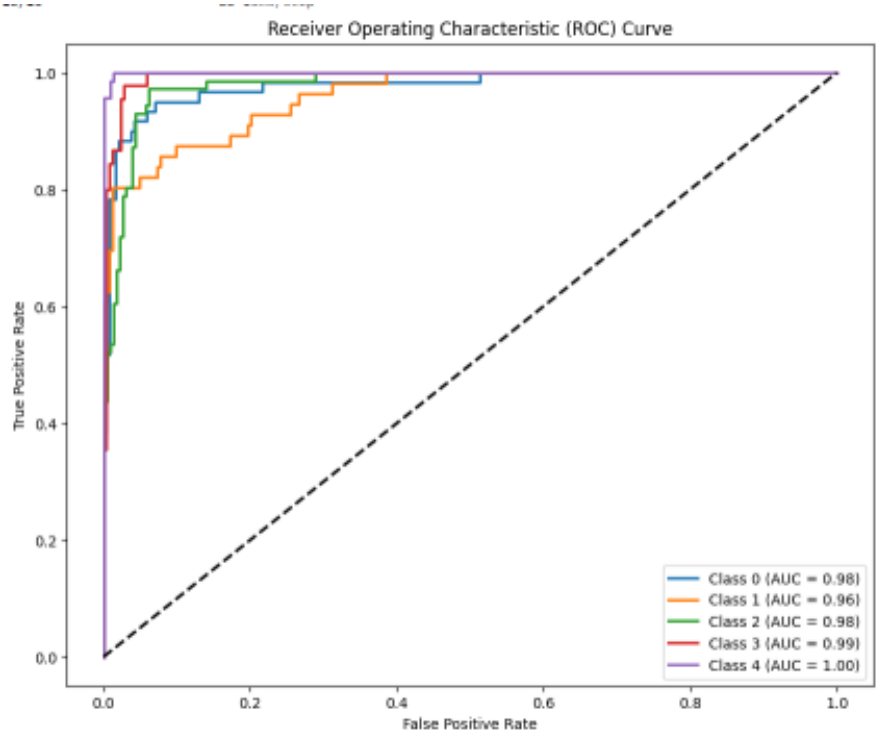
Loss Curve



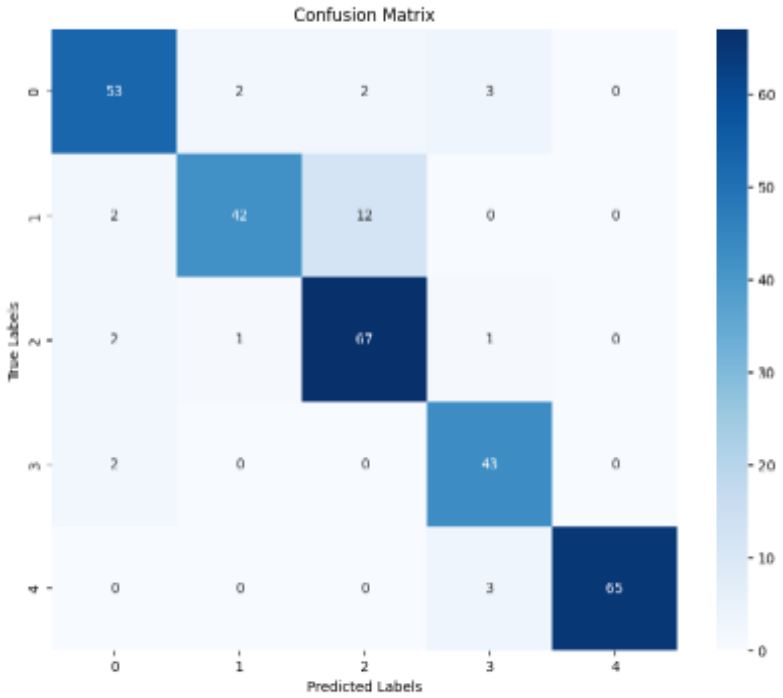
Accuracy Curve



ROC Curve



Confusion Matrix



Classification Report

Class	Precision	Recall	F1-Score	Support
Foggy	0.8983	0.8833	0.8908	60
Rainy	0.9333	0.7500	0.8317	56
Cloudy	0.8272	0.9437	0.8816	71
Sunrise	0.8600	0.9556	0.9053	45
Shine	1.0000	0.9559	0.9774	68
Accuracy			**0.9000**	300
Macro Avg	0.9038	0.8977	0.8973	300
Weighted Avg	0.9053	0.9000	0.8994	300