Project Presentation

Project Overview

- *Idea : Weather Classification Description*
- *Goal: Classify weather conditions based on images*
- *Dataset:Weather Image Recognition*
- *Models:*

  1) ResNet

  2) MobileNet

  3) VGG19

  4) Inception V1

Dataset Description

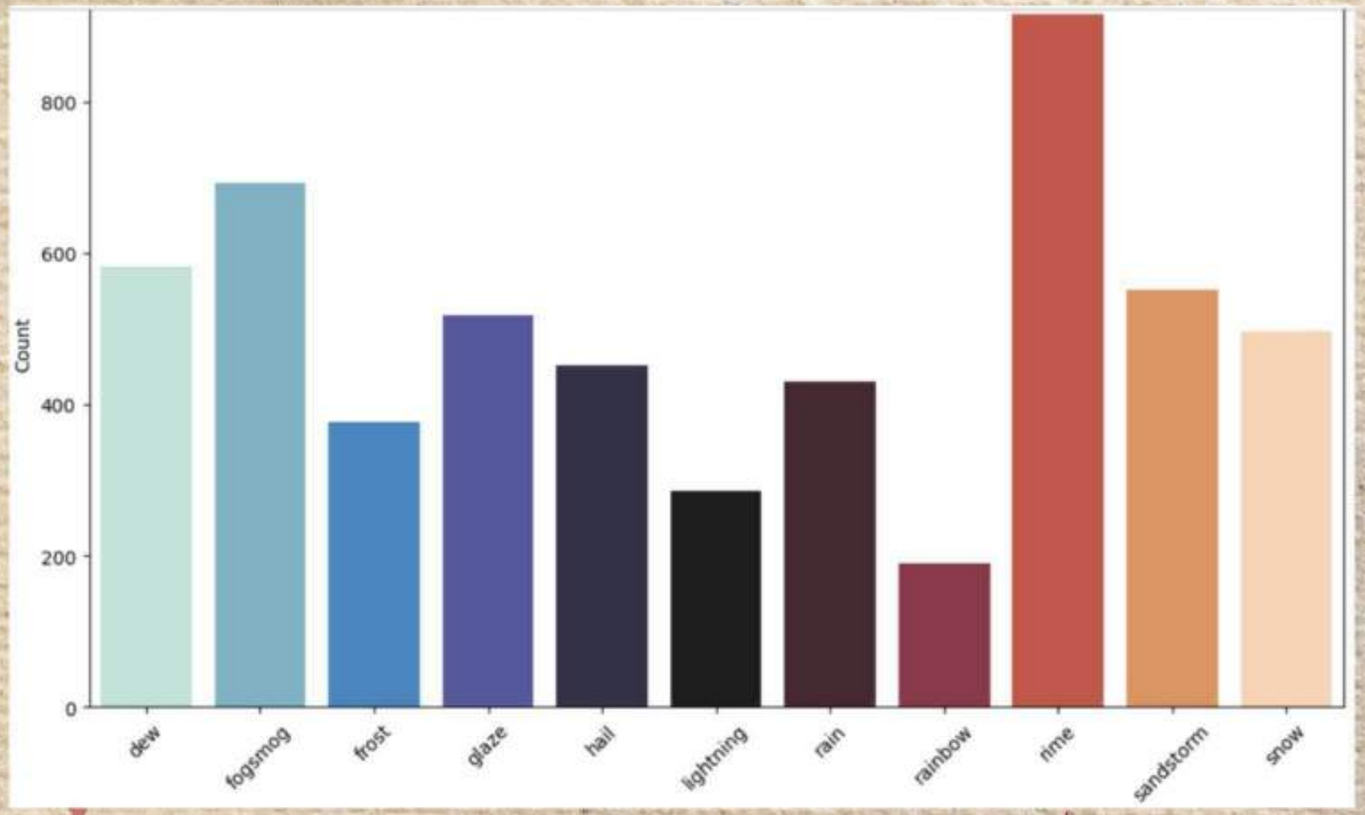
- *The dataset consists of 11 weather-related classes, with a total of 6,862 images distributed unevenly across categories.*

- *Detailed distribution:*

```
Number of images in dew: 698
First 5 files in dew: ['2208.jpg', '2209.jpg', '2210.jpg', '2211.jpg', '2212.jpg']
Number of images in fogsmog: 851
First 5 files in fogsmog: ['4075.jpg', '4076.jpg', '4077.jpg', '4078.jpg', '4079.jpg']
Number of images in frost: 475
First 5 files in frost: ['3600.jpg', '3601.jpg', '3602.jpg', '3603.jpg', '3604.jpg']
Number of images in glaze: 639
First 5 files in glaze: ['6090.jpg', '6091.jpg', '6092.jpg', '6093.jpg', '6094.jpg']
Number of images in hail: 591
First 5 files in hail: ['0000.jpg', '0001.jpg', '0002.jpg', '0003.jpg', '0004.jpg']
Number of images in lightning: 377
First 5 files in lightning: ['1830.jpg', '1831.jpg', '1832.jpg', '1833.jpg', '1834.jpg']
Number of images in rain: 526
First 5 files in rain: ['1011.jpg', '1013.jpg', '1017.jpg', '102.jpg', '1021.jpg']
Number of images in rainbow: 232
First 5 files in rainbow: ['0592.jpg', '0593.jpg', '0594.jpg', '0595.jpg', '0596.jpg']
Number of images in rime: 1160
First 5 files in rime: ['4930.jpg', '4931.jpg', '4932.jpg', '4933.jpg', '4934.jpg']
Number of images in sandstorm: 692
First 5 files in sandstorm: ['2908.jpg', '2909.jpg', '2910.jpg', '2911.jpg', '2912.jpg']
Number of images in snow: 621
First 5 files in snow: ['0830.jpg', '0831.jpg', '0832.jpg', '0833.jpg', '0834.jpg']
```

Dataset Description

- *Data Imbalanced:*



- *So, We applied Data augmentation*

```
def augment_and_preprocess(image, label,pre):
    image = flip_layer(image)
    image = rotation_layer(image)
    image = translation_layer(image)
    image = zoom_layer(image)
    image = pre(image)
    return image, label
```

MobileNet OverView

- *Architecture: MobileNet is a family of convolutional neural network (CNN) architectures designed for image classification, object detection, and other computer vision tasks. They are designed for small size, low latency, and low power consumption, making them suitable for on-device inference and edge computing on resource-constrained devices like mobile phones and embedded systems. They were originally designed to be run efficiently on mobile devices with TensorFlow Lite.*

https://en.wikipedia.org/wiki/MobileNet

MobileNet OverView

- *Paper: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*
- *Architecture: MobileNet is built entirely on Depthwise Separable Convolutions, which replace the standard convolution to reduce computation and model size.*
- *Structure: MobileNet is organized as a series of repeated convolutional blocks, gradually reducing the spatial dimensions and increasing the number of channels*
  - *1 Standard Conv layer at the beginning*
  - *13 Depthwise Separable Conv blocks → 13 × 2 = 26 layers*
  - *Average Pooling + Fully Connected layer = 1 layer*

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

https://arxiv.org/pdf/1704.04861

Our MobileNet Model

- *In this project, MobileNet is used as a feature extractor with fine-tuning to adapt it to a custom dataset containing 11 classes.*

Step 1: Input Layer

- *Input images are resized to 150×150 pixels and preprocessed using the MobileNet _preprocess _input to normalize pixel values.*

Step 2: Pre-trained MobileNet Backbone

- *include_top = False*
- *weights = ImageNet*
- *pooling = 'avg'*

The top classification layers of MobileNet are removed, allowing the model to act as a feature extractor.

Global average pooling is applied to reduce spatial dimensions while preserving semantic information

Our MobileNet Model

Step 3: Transfer Learning

- ***To prevent overfitting and reduce training time, the early layers of MobileNet are frozen. The last 30 layers are fine-tuned to better adapt high-level features to the target dataset.***

Step 4: Fully Connected Layers

- ***A fully connected layer with 512 neurons and ReLU activation is added to learn task-specific features, followed by a dropout layer with a rate of 0.5 to reduce overfitting.***

Step 5: Output Layer

- ***The final softmax layer outputs probability distributions over the 11 target classes.***

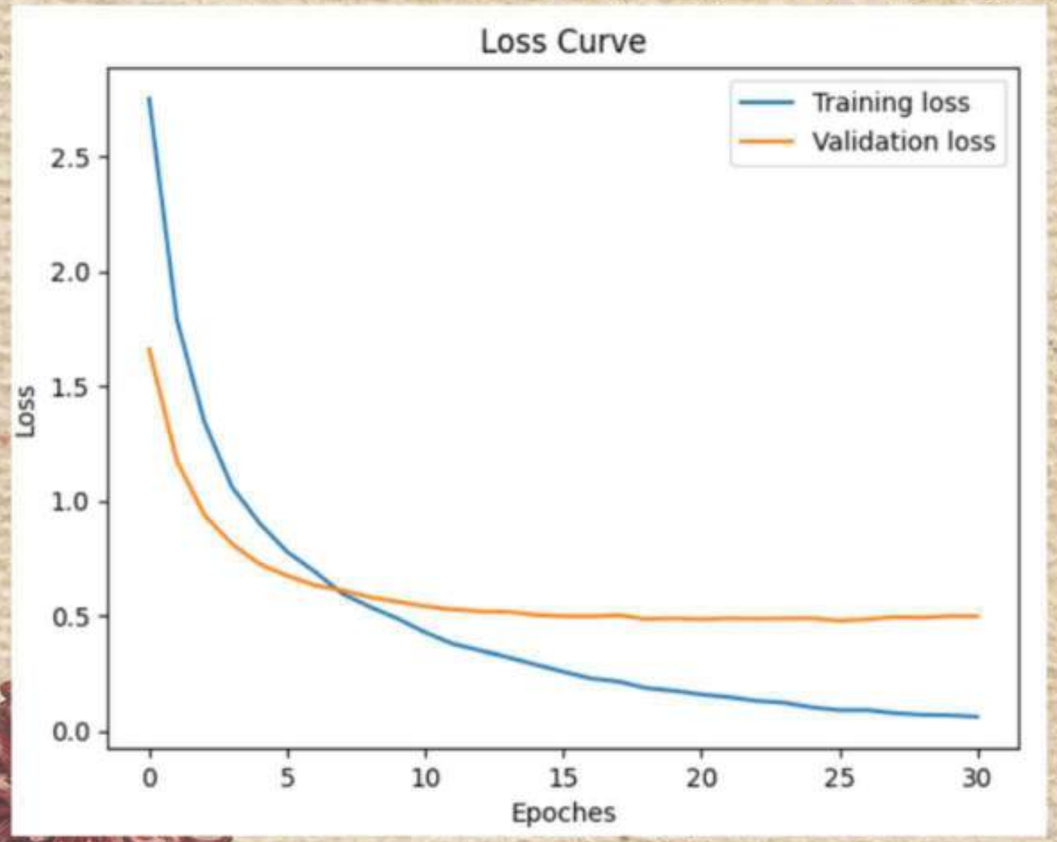
Our MobileNet Model

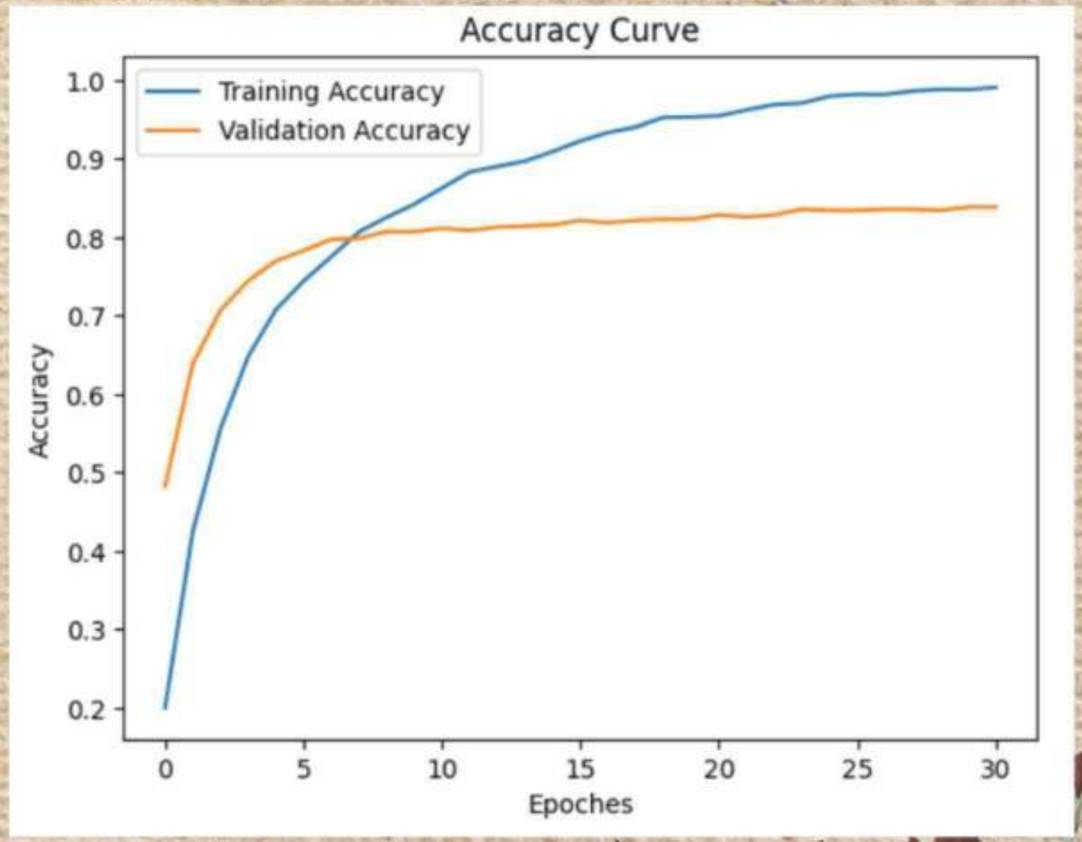
Training Configuration:

- ***Optimizer-> Adam***
- ***Learning Rate->0.00001***
- ***Epochs-> 50***
- ***Early Stopping-> Patience = 5***
-

The model was trained using the Adam optimizer with a low learning rate to ensure stable fine-tuning of pre-trained weights.

Our MobileNet Model





At Epoch 31:

- ***Training Accuracy: 99.03%***
- ***Validation Accuracy: 83.86%***
- ***Validation Loss: 0.49***

- ***While the model achieves very high training accuracy, a noticeable gap exists between training and validation performance, indicating mild overfitting.***

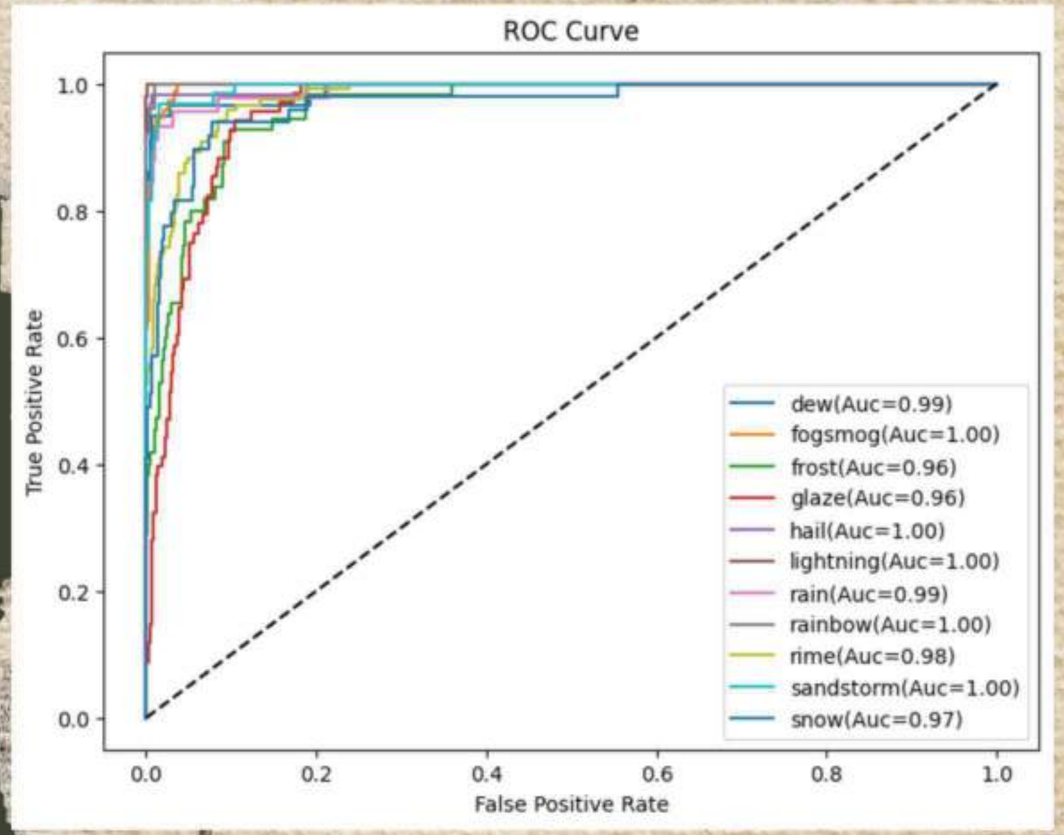
Our MobileNet Model

```
accuracy, precision, recall, f1 = Calculate_Confusion_Matrix(y_true,y_pred)
print ("accuracy =", accuracy)
print ("precision =", precision)
print ("recall =", recall)
print ("f1 =" ,f1)
```

```
accuracy = 0.8273809523809523
precision = 0.8419127693052574
recall = 0.8295217813591134
f1 = 0.8322518010639833
```

- *Out of every 100 images in the test→ The model correctly rated approximately 83 images*
- *A high precision value shows that the model makes reliable predictions with a low rate of false positives.*
- *The recall score indicates the model's ability to correctly identify most instances of each class(False Negatives are not high).*
- *The F1-score confirms a balanced trade-off between precision and recall, which is essential for multi-classclassification tasks.*

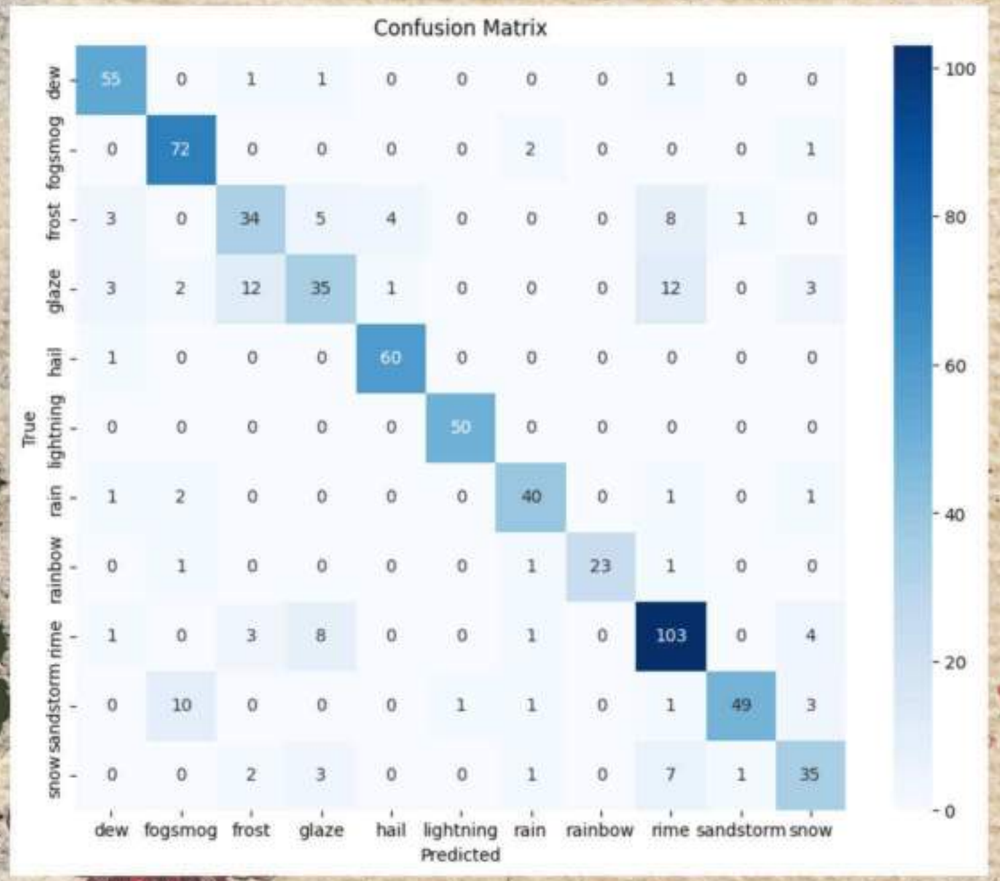
Our MobileNet Model



- *Classes:fogsmog, hail, lightning, rainbow, sandstorm (AUC = 1.00): Classes with distinct visual characteristics achieve perfect AUC scores, indicating near-perfect separability from other weather conditions.*

- *Classes: frost (0.96), glaze (0.96), snow (0.97) These classes exhibit overlapping texture and color patterns, making class boundaries less distinct and vincreasing the likelihood of misclassification.*

Our MobileNet Model



- *"The model demonstrated high power in classifying certain categories, with a very high percentage of correct classifications. The most notable of these categories are:*
- *Rime (103): This is the most accurate classification.*
- *Fogsfog (72).*
- *Hail (60).*
- *Lightning (50)."*

- *"The model demonstrated high power in classifying certain categories, with a very high percentage of correct classifications. The most notable of these categories are:*
- *Rime (103): This is the most accurate classification.*
- *Fogsfog (72).*
- *Hail (60).*
- *Lightning (50)."*

ResNet50 OverView

- ***ResNet-50 is a popular deep-learning image classification model. Version 1.5 of the Residual Neural Networks family of models, ResNet-50 is a 50-layer convolutional neural network (CNN).***

https://blog.roboflow.com/what-is-resnet-50/

Stacking the Blocks: Building ResNet-50

ResNet-50 incorporates 50 bottleneck residual blocks, arranged in a stacked manner. The early layers of the network feature conventional convolutional and pooling layers to preprocess the image before it undergoes further processing by the residual blocks. Ultimately, fully connected layers positioned at the pinnacle of the structure utilize the refined data to categorize the image with precision.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

ResNet Architectures. Building blocks are shown in brackets with the number of blocks stacked. ResNet-50

ResNet50 OverView

- ***ResNet-50 is a popular deep-learning image classification model. Version 1.5 of the Residual Neural Networks family of models, ResNet-50 is a 50-layer convolutional neural network (CNN).***
- ***Designed to solve the problem of:Vanishing Gradient in Deep Networks***

https://blog.roboflow.com/what-is-resnet-50/

Stacking the Blocks: Building ResNet-50

ResNet-50 incorporates 50 bottleneck residual blocks, arranged in a stacked manner. The early layers of the network feature conventional convolutional and pooling layers to preprocess the image before it undergoes further processing by the residual blocks. Ultimately, fully connected layers positioned at the pinnacle of the structure utilize the refined data to categorize the image with precision.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	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

ResNet Architectures. Building blocks are shown in brackets with the number of blocks stacked. ResNet-50

Our ResNet50 Model

Step 1: Input Layer
Input images are resized to 150×150 pixels and preprocessed using the resnet50 _preprocess _input to match the distribution of the ImageNet dataset.

Step 2: Pre-trained ResNet50 Backbone

- ***include_top = False***
- ***weights = ImageNet***
- ***pooling ='avg'***

The top classification layers are removed, allowing ResNet50 to extract high-level visual features from the input images.

Our ResNet50 Model

Step 3: Transfer Learning

- ***To balance generalization and task-specific learning, the majority of layers are frozen while the last 30 layers are fine-tuned to adapt high-level features to the target dataset***

Step 4: Fully Connected Layers

- ***A fully connected layer with 512 neurons and ReLU activation is added to learn task-specific features, followed by a dropout layer with a rate of 0.5 to reduce overfitting.***

Step 5: Output Layer

- ***The final softmax layer outputs probability distributions over the 11 target classes.***
- ***L2 regularization is applied to the output layer to further reduce overfitting and improve generalization.***

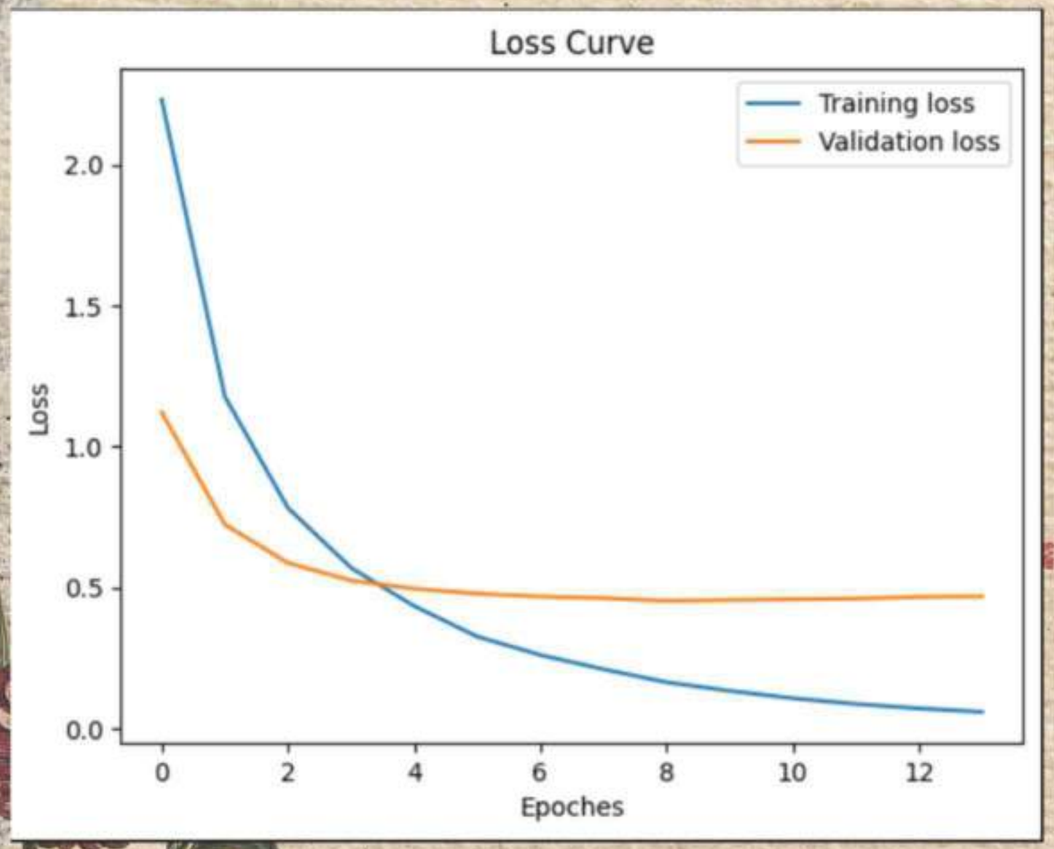
Our ResNet50 Model

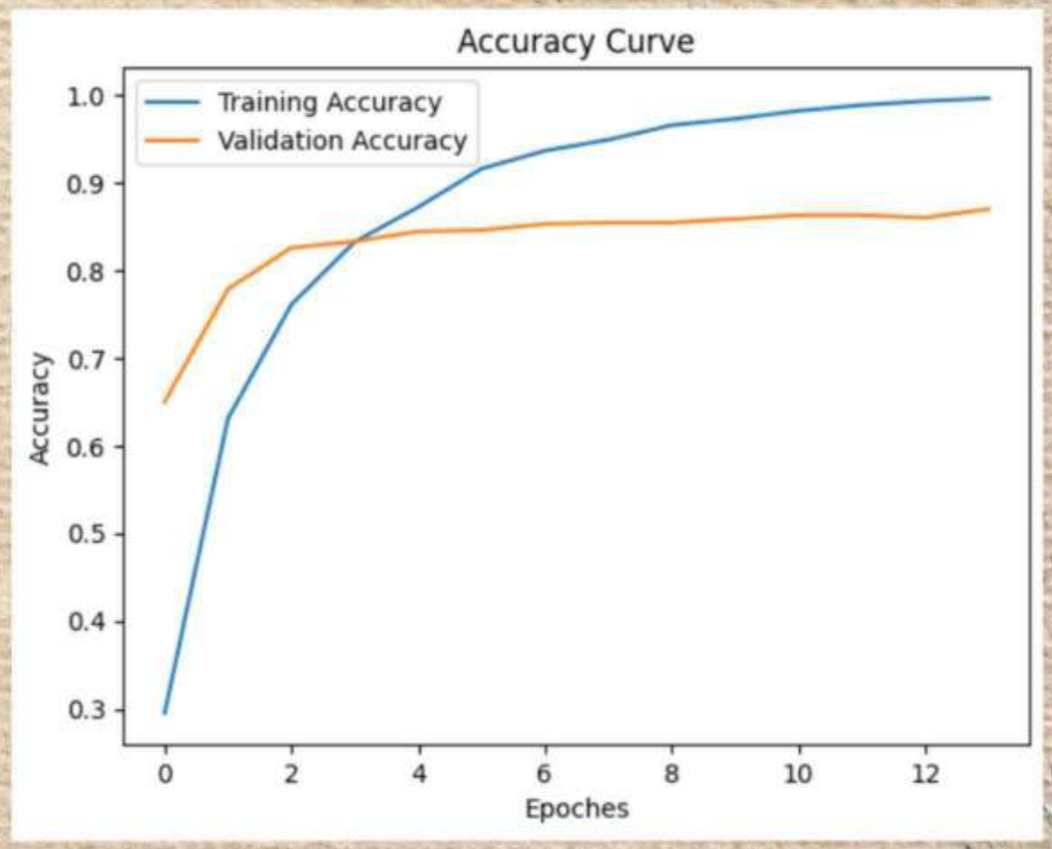
Training Configuration:

- ***Optimizer-> Adam***
- ***Learning Rate->0.00001***
- ***Epochs-> 50***
- ***Early Stopping-> Patience = 5***
-

The model was trained using the Adam optimizer with a low learning rate to ensure stable fine-tuning of pre-trained weights.

Our ResNet50 Model





At Epoch 14:

- ***Training Accuracy: 99.64%***
- ***Validation Accuracy: 87.00%***
- ***Validation Loss: 0.469***

- ***ResNet50 achieves very high training accuracy while maintaining strong validation performance, indicating improved generalization compared to lighter architectures.***

Our ResNet50 Model

```
accuracy, precision, recall, f1 = Calculate_Confusion_Matrix(y_true,y_pred)
print ("accuracy =", accuracy)
print ("precision =", precision)
print ("recall =", recall)
print ("f1 =" ,f1)
```

```
accuracy = 0.8556547619047619
precision = 0.872768614547569
recall = 0.8517384541644062
f1 = 0.8594783458064886
```

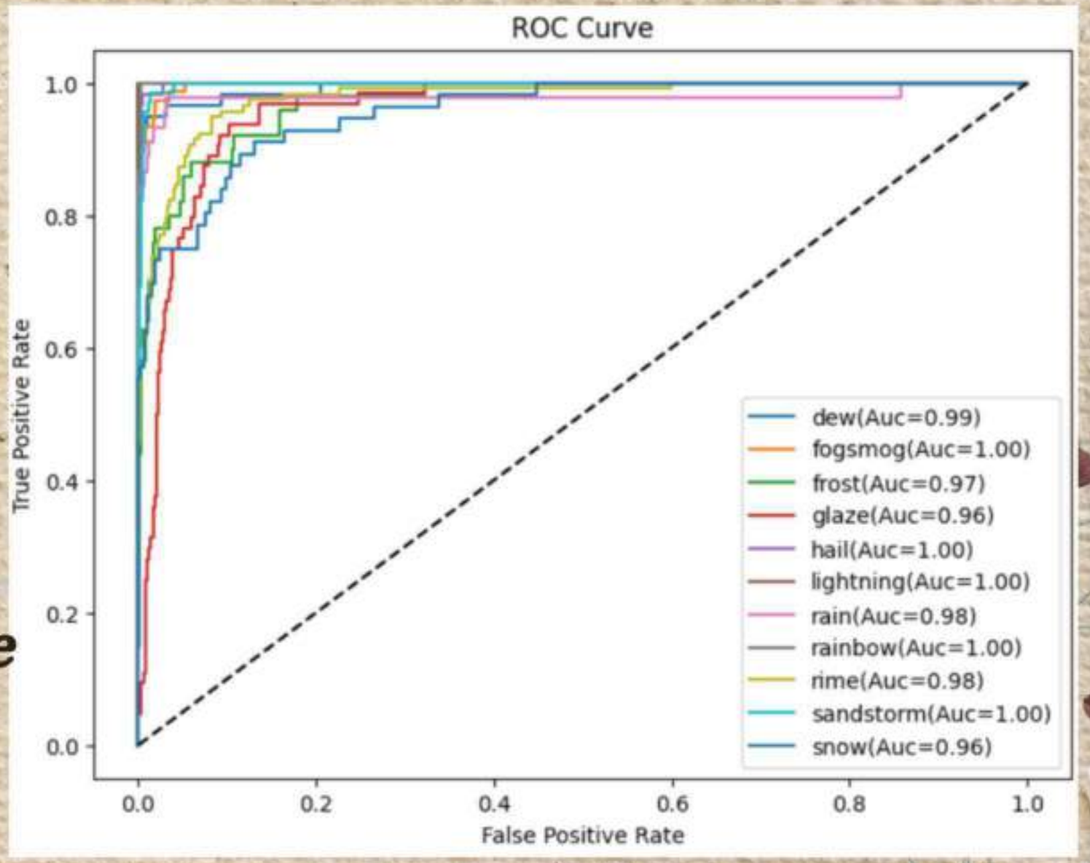
- ***Accuracy = 85.57%: Out of every 100 test images:→ The model correctly categorized approximately 86 images Higher than MobileNet→ Better generalization capabilities.***
- ***A high precision value demonstrates that the model produces reliable predictions with minimal false alarms(The number of False Positives is small).***
- ***The recall score reflects the model's ability to correctly identify most true instances across all classes.***
- ***The F1-score confirms a well-balanced model that maintains strong performance without favoring precision over recall or vice versa.***

Our ResNet50 Model

- *Classes with near-perfect performance (AUC ≈ 1.00)*
- *fogsmog*
- *hail*
- *lightning*
- *rainbow*
- *sandstorm*

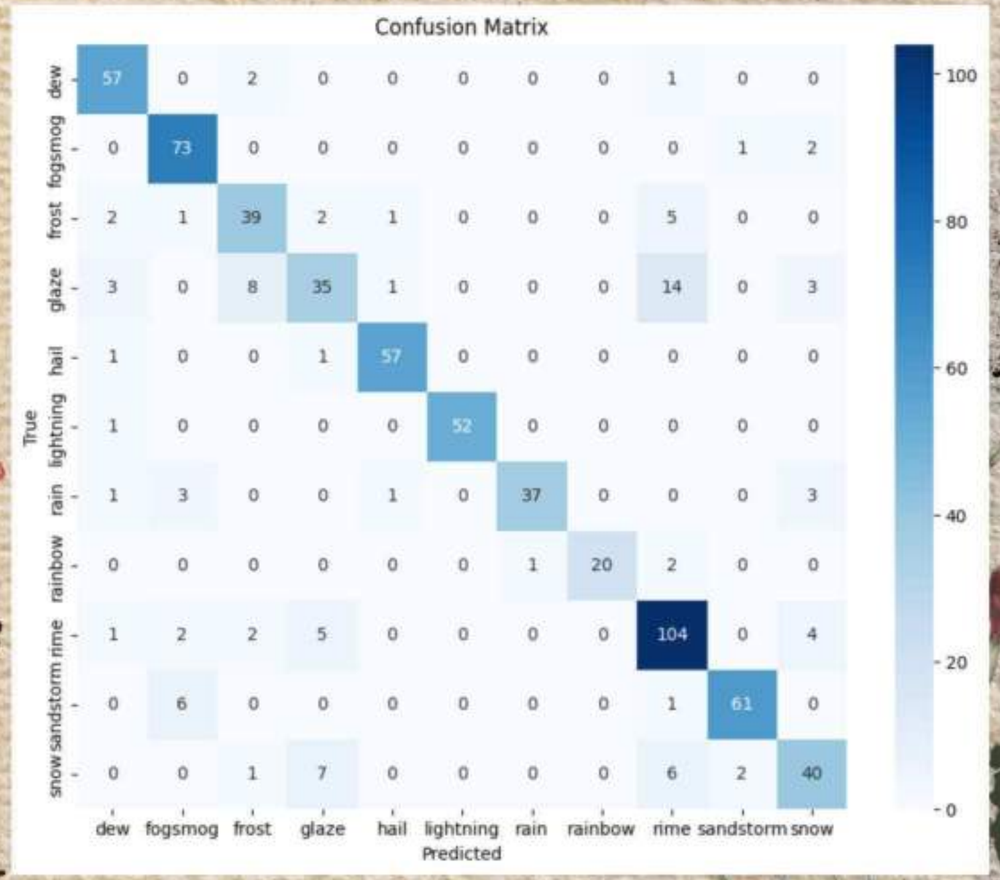
- *Classes with very strong performance (AUC = 0.98 – 0.99)*
- *dew (0.99)*
- *rain (0.98)*
- *rime (0.98)*

- *Slightly similar to other Classes, but ResNet managed to reduce errors compared to MobileNet*



Our ResNet50 Model

- *Classes with near-perfect performance (AUC ≈ 1.00)*
- *fogsmog*
- *hail*
- *lightning*
- *rainbow*
- *sandstorm*



- *Frost: This class was the weakest, with 12 samples incorrectly classified as Glaze and 8 as Rime, compared to only 3 correct classifications.*
- *Glaze: The model suffered from confusion between Glaze, Frost, and Rime, with 12 Glaze samples incorrectly classified as Rime and 12 as Frost.*
- *Sandstorm: A significant number of Sandstorm samples (10) were confused with Fogsfog.*

VGG19 OverView

- *VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.*

- *https://www.mathworks.com/help/deeplearning/ref/vgg19.html*

Our VGG19 Model

- *Step 1: Input Layer*
  Input images are resized to 150×150 pixels .

- *Step 2: 5 Convolutional Blocks*
  Block 1 :
  ** Conv2D(32) → BatchNorm → Conv2D(32) → BatchNorm → MaxPooling*
  **It starts showing simple features:*
  Edges
  Corners
  Lines
  Number of filters = 32
  Filter size = 3x3

**MaxPooling:*
Reduces image size for text
From 150x150 → 75x75
Reduces calculations and preserves the most important information

Our VGG19 Model

Step 2: 5 Convolutional Blocks

Block 2 :

**** Conv2D(64) → BatchNorm → Conv2D(64) → BatchNorm → MaxPooling***

****Learn more details:***

Shapes

Patterns

Number of filters increased to 64

Size decreased:

75×75 → 37×37

Step 2: 5 Convolutional Blocks

Block 3 :

**** Conv2D(128) → BatchNorm → Conv2D(128) → BatchNorm → MaxPooling***

****It begins to understand the parts of an object, such as:***

Part of a face

Part of a body

Number of filters = 128

Size:

37×37 → 18×18

Our VGG19 Model

- ***Step 2: 5 Convolutional Blocks***
  Block 4 :
  **** Conv2D(256) → BatchNorm → Conv2D(256) → BatchNorm → MaxPooling***
  ****High-level features***
  The model now almost "understands the image"
  Size:
  18×18 → 9×9

- ***Step 2: 5 Convolutional Blocks***
  Block 5 :
  **** Conv2D(256) → BatchNorm → Conv2D(256) → BatchNorm → MaxPooling***
  ****Highest level of understanding***
  Final size:
  9×9 → 4×4

Our VGG19 Model

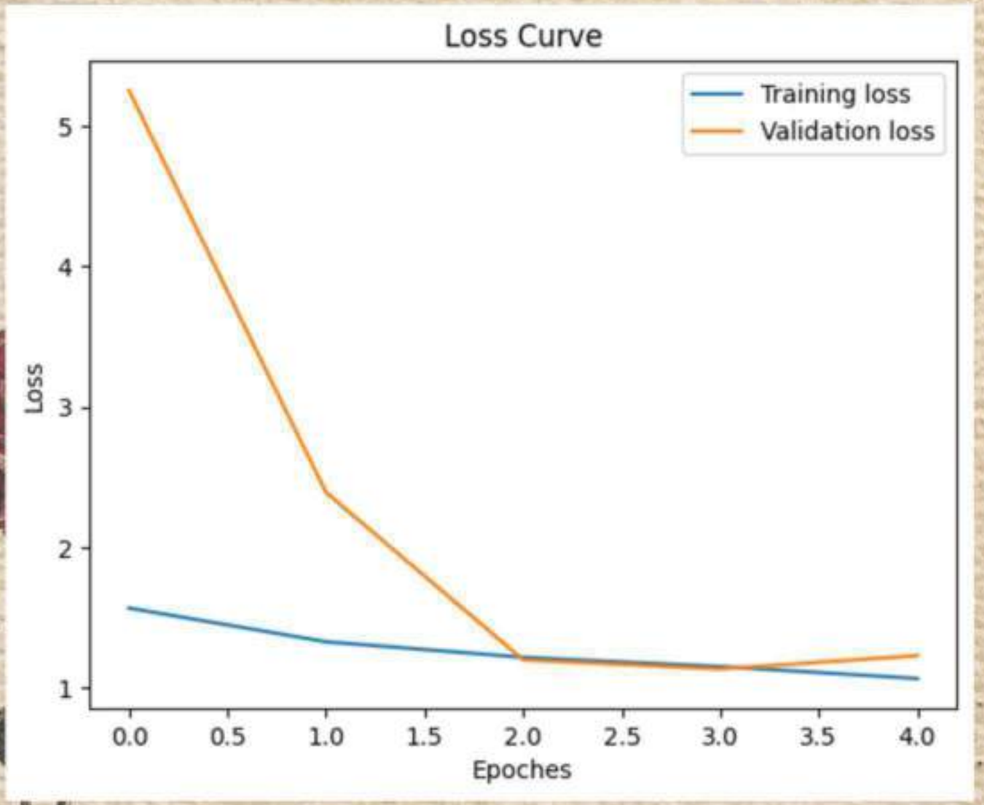
Fully Connected Layers

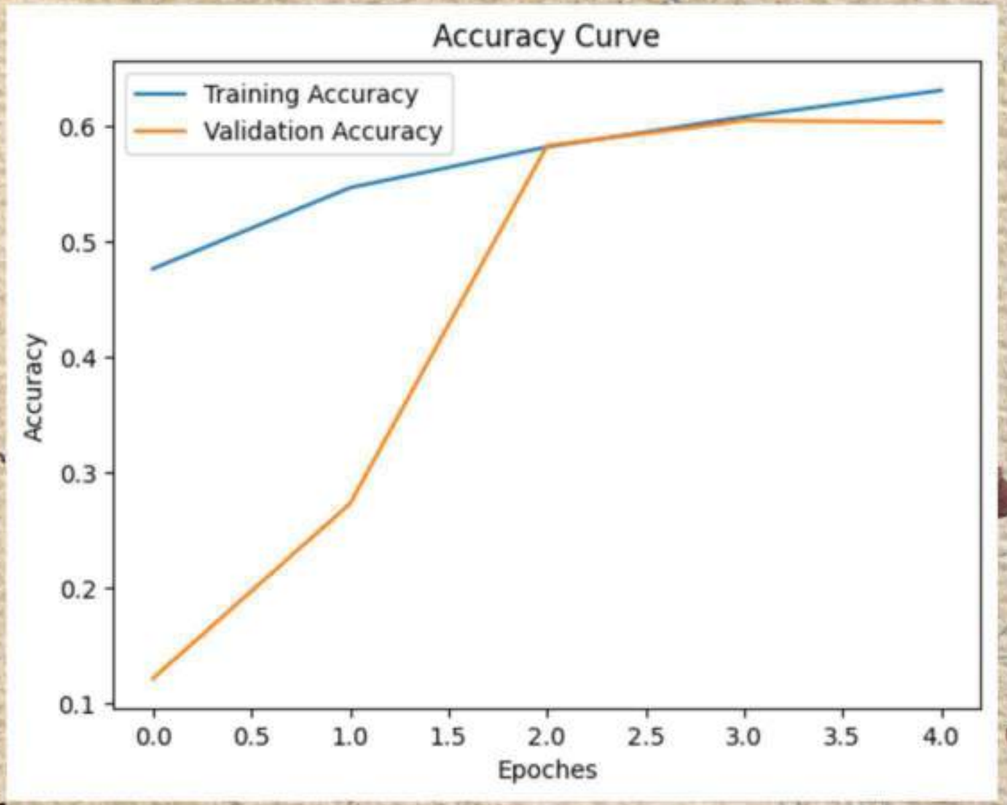
- *A fully connected layer with 512 neurons and ReLU activation is added to learn task-specific features, followed by a dropout layer with a rate of 0.5 to reduce overfitting.*

Output Layer

- *The final softmax layer outputs probability distributions over the 11 target classes.*

Our VGG19 Model





At Epoch 5:

Training Accuracy: ~63%

Training Loss: ~1.06

Validation Accuracy: ~60%

Validation Loss: ~1.23

****The difference between Training and Validation isn't significant.***

****There's no major overfitting.***

Our VGG19 Model

- *The Accuracy value indicates that the model was able to correctly classify approximately 57% of the images.*
- *The Recall value is slightly lower than the Precision value, which means:*
  The model sometimes fails to detect all samples belonging to the correct category.
- *Some correct images are not detected (False Negatives).*
- *The F1-score highlights the trade-off between precision and recall and confirms that the overall performance is moderate.*

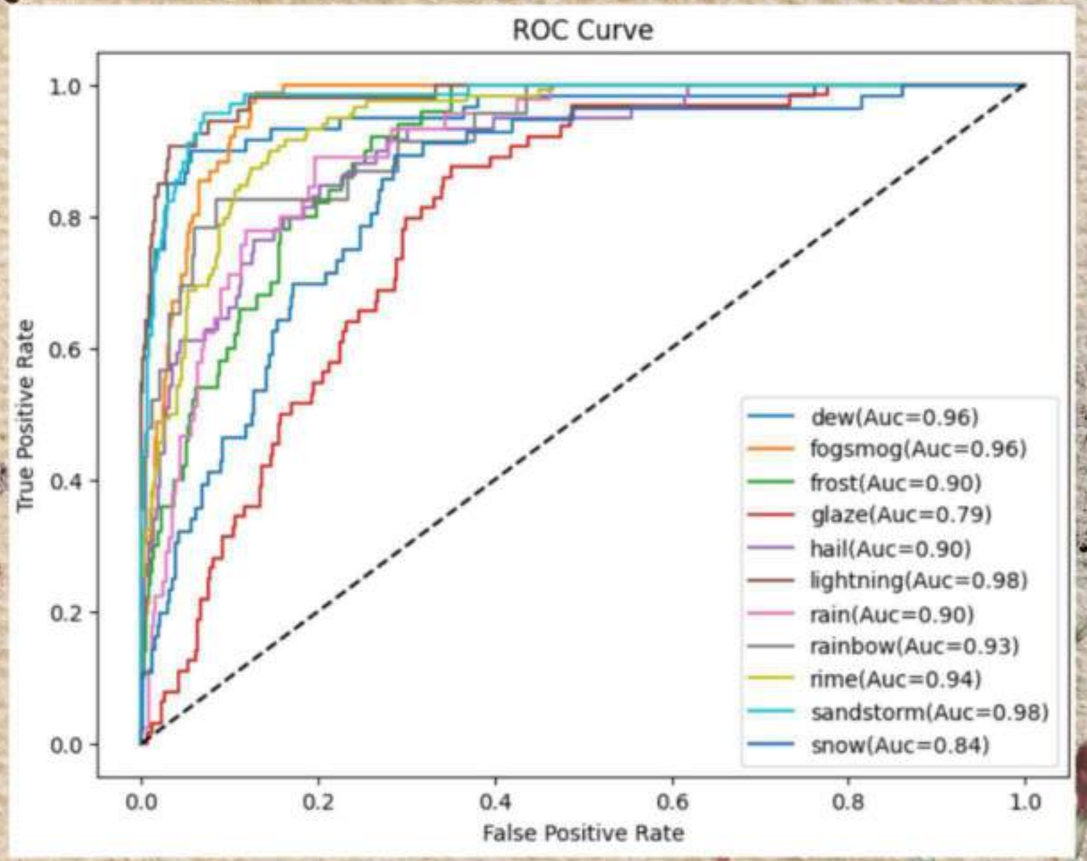
```
# --- Evaluate with Helper Functions ---

accuracy, precision, recall, f1 = Calculate_Confusion_Matrix(y_true, y_pred)

print("\n--- Test Metrics ---")
print("Accuracy =", accuracy)
print("Precision =", precision)
print("Recall =", recall)
print("F1 Score =", f1)


class_names = train_dataset.class_names

Plot_Confuision_Matrix(y_true, y_pred, class_names)
Plot_Roc_Auc_Curve(y_true, y_score, class_names)
```

```
--- Test Metrics ---
Accuracy = 0.5699404761904762
Precision = 0.5674039239655845
Recall = 0.5436978256389693
F1 Score = 0.543870424536802
```

- *The Precision value is close to Accuracy, indicating that:*
  The model does not significantly overestimate the number of incorrect predictions for the classes.
- *The percentage of False Positives is not very high.*

Our VGG19 Model

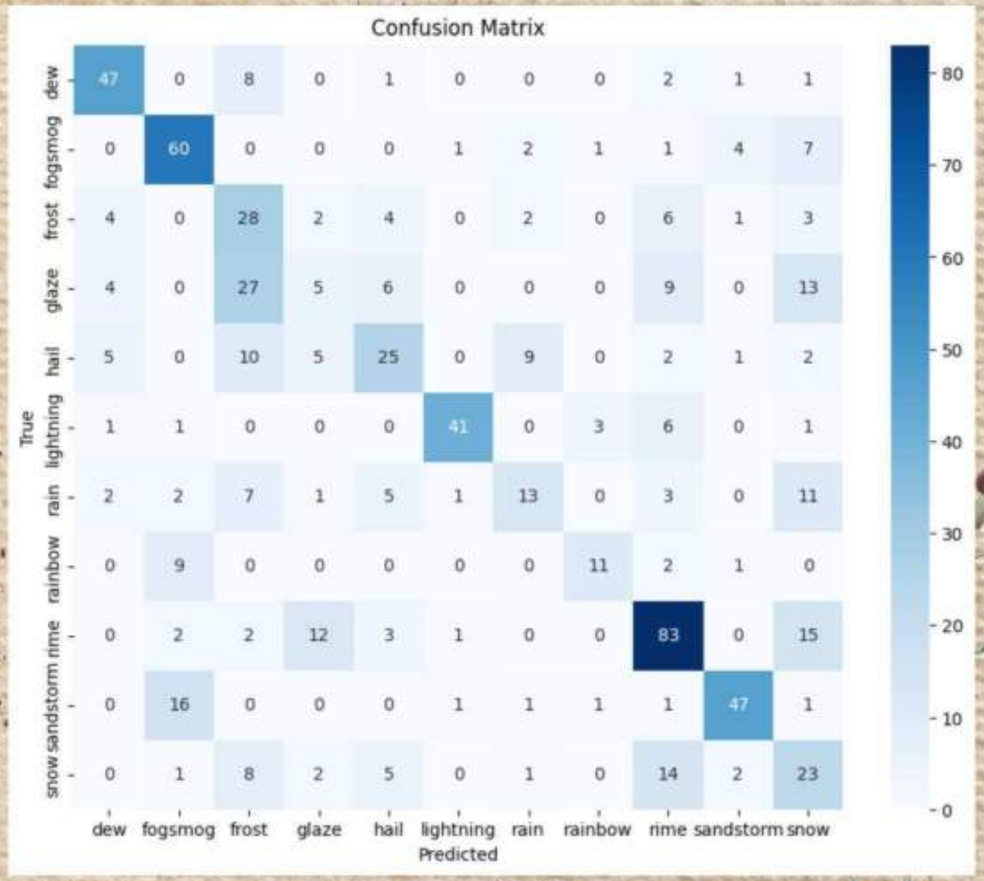
- *Strongest Categories (AUC ~ 0.96): The lightning (0.98) and sandstorm (0.98) categories show the best performance. This means the model is excellent at distinguishing these phenomena from other categories. Their curves are located almost in the upper left corner, indicating very high correct detection rates and low false alarm rates.*



- *Weakest Categories (AUC ~ 0.84): Glaze (0.79): This is the weakest performance recorded. The low AUC value indicates that the model has significant difficulty separating the "Glaze" category from the others. This curve (red line) is the closest to the dashed line (random guess) among all the curves. Snow (0.84): Although acceptable, it is lower than most of the other categories, indicating discrimination challenges.*

Our VGG19 Model

- ***Rime (83) and Fogsfog (60) are the most accurate in VGG19.***
- ***Categories such as Rain (13), Rainbow (11), Hail (25), and Glaze (27) have significantly fewer correct classifications than Rime.***



Inception_V1 OverView

- *Inception[1] is a family of convolutional neural network (CNN) for computer vision, introduced by researchers at Google in 2014 as GoogLeNet (later renamed Inception v1). The series was historically important as an early CNN that separates the stem (data ingest), body (data processing), and head (prediction), an architectural design that persists in all modern CNN.[2]*

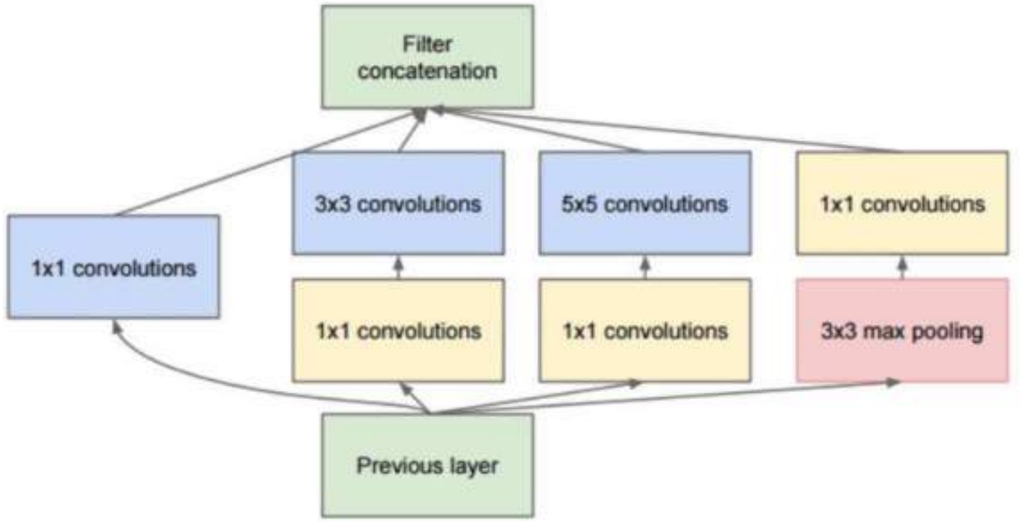
- *Inception v1 is deep, it suffered from the vanishing gradient problem. The team solved it by using two "auxiliary classifiers", which are linear-softmax classifiers inserted at 1/3-deep and 2/3-deep within the network, and the loss function is a weighted sum of all three:L=0.3Laux,1+0.3Laux,2+Lreal*
- *These were removed after training was complete. This was later solved by the ResNet architecture.*

Inception_V1 OverView

Inception v1

Going Deeper with Convolutions (ILSVRC 2014)

By adding auxiliary classifiers connected to these intermediate layers, we would expect to encourage discrimination in the lower stages in the classifier, increase the gradient signal that gets propagated back, and provide additional regularization.



Application: GoogLeNet

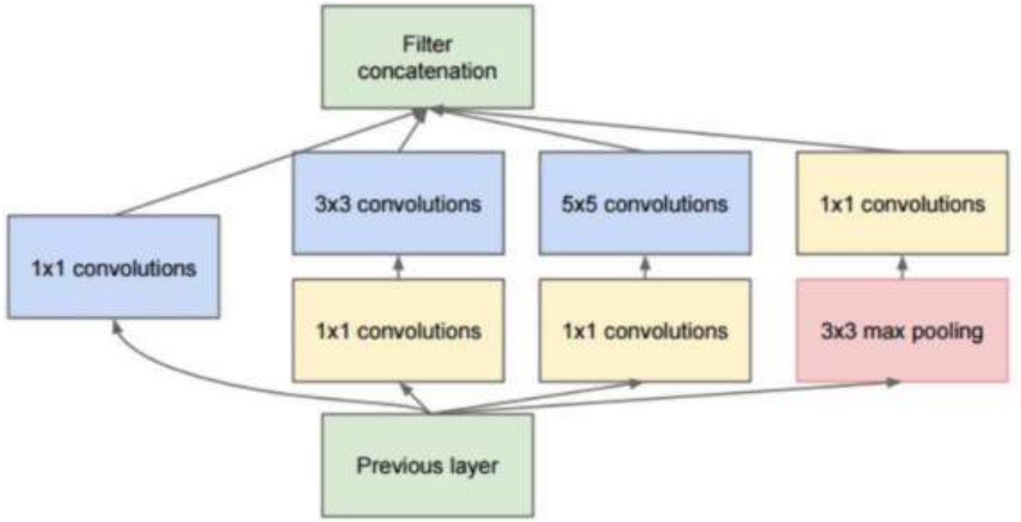
- https://en.wikipedia.org/wiki/Inception_%28deep_learning_architecture%29?utm_source=chatgpt.com
- https://machine-learning-note.readthedocs.io/en/latest/CNN/inception.html

Inception_V1 OverView

Inception v1

Going Deeper with Convolutions (ILSVRC 2014)

By adding auxiliary classifiers connected to these intermediate layers, we would expect to encourage discrimination in the lower stages in the classifier, increase the gradient signal that gets propagated back, and provide additional regularization.



Application: GoogLeNet

- https://en.wikipedia.org/wiki/Inception_%28deep_learning_architecture%29?utm_source=chatgpt.com
- https://machine-learning-note.readthedocs.io/en/latest/CNN/inception.html

OurInception_V1 Model

Preprocessing Layers (Outside the Model)
Before images are fed into the grid, the following processing steps are applied:

Training Data
Resize to (224 × 224)
Random Horizontal Flip
Convert Image to Tensor
Normalization using ImageNet values:
Mean = [0.485, 0.456, 0.406]
Std = [0.229, 0.224, 0.225]
Validation & Test Data
Resize to (224 × 224)
Convert to Tensor
Normalization

OurInception_V1 Model

The data was divided into:
70% Training
15% Validation
15% Testing

Image Preprocessing
Image sizes have been standardized to:
224 × 224 × 3

Step 1: Input Layer
224 × 224 × 3

OurInception_V1 Model

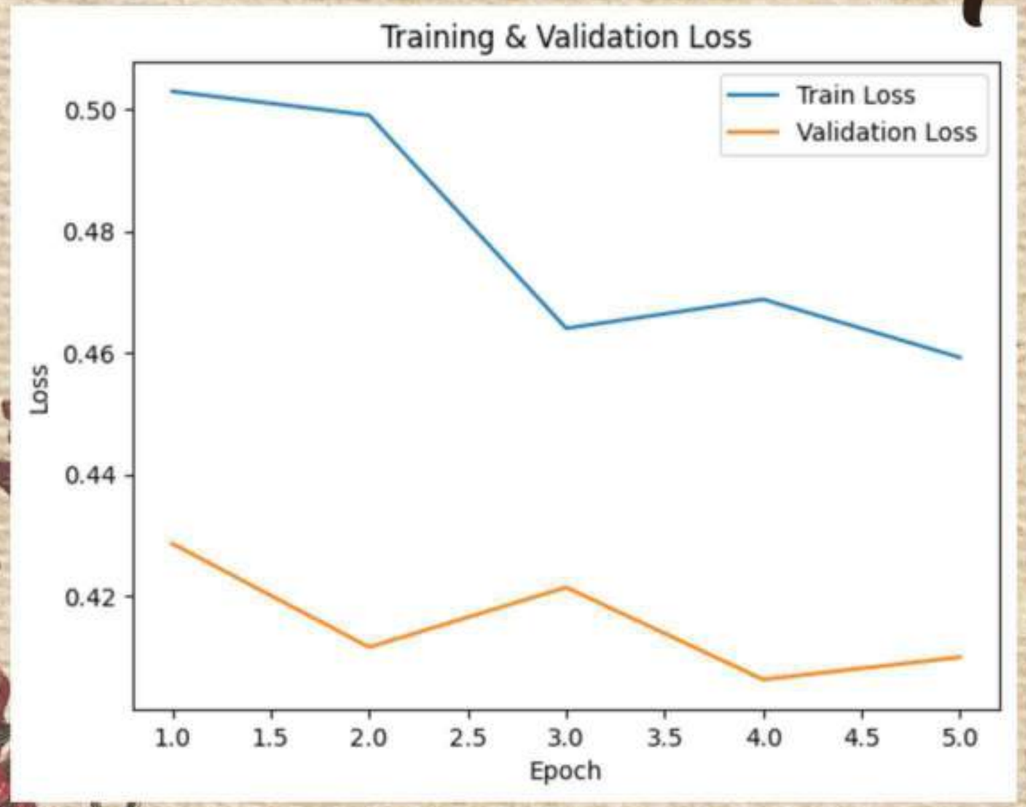
- *Inception v1 is built with stacked Inception modules instead of simple Conv2D layers.*
- *Each Inception module contains parallel branches:*
  a. *1×1 Convolution → reduces depth and extracts simple features*
  b. *3×3 Convolution → captures medium-scale features*
  c. *5×5 Convolution → captures large-scale features*
  d. *3×3 MaxPooling + 1×1 Convolution → preserves important information and reduces feature map size*

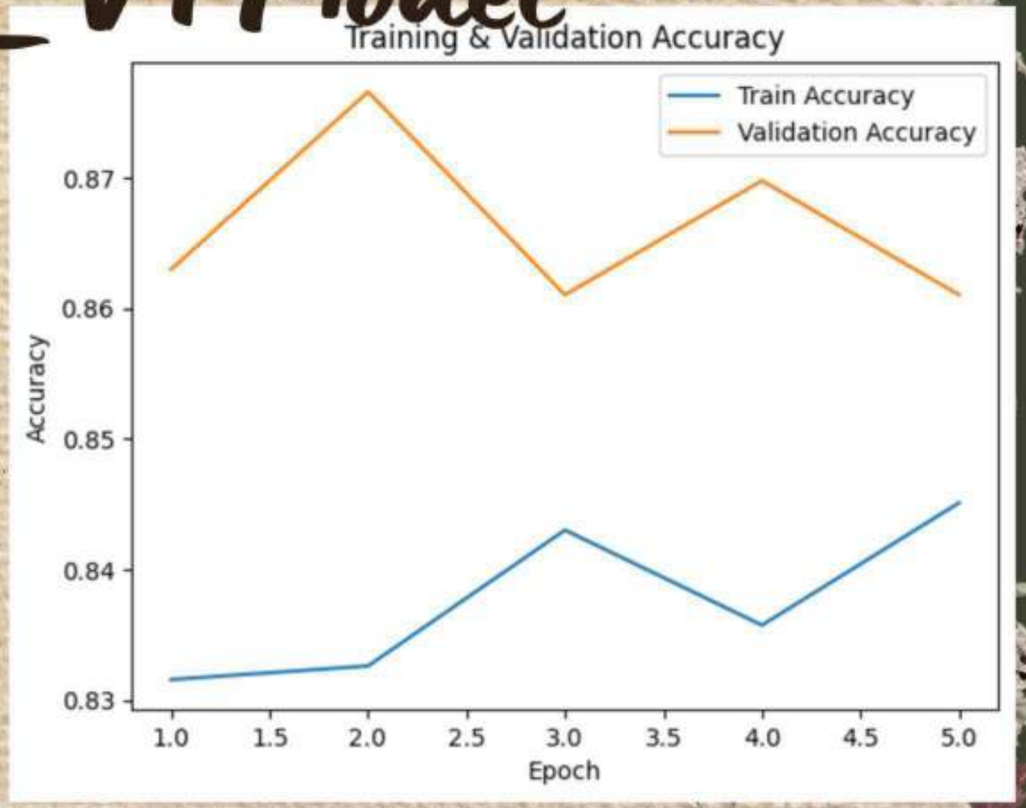
- *Inception v1 includes auxiliary classifiers during training to help reduce the vanishing gradient problem.*
- *During inference, only the main classifier is used.*
- *ReLU is applied after every convolution in the Inception modules.*
-

OurInception_V1 Model

- *The final fully connected layer aggregates all learned features.*
- *Outputs a vector of length NUM_CLASSES = 11 representing class scores.*
- *During training, CrossEntropyLoss applies Softmax internally.*

OurInception_V1 Model





At Epoch 5:

- ***Train Accuracy = 84.5%***
- ***Validation Accuracy = 86.1%***
- ***Train Loss = 0.459***
- ***Validation Loss = 0.410***

****Validation loss is slightly lower than training loss → no overfitting detected***

OurInception_V1 Model

- *High performance: dew, hail, lightning, rainbow (F1 > 0.90)*
- *Lower performance: frost and snow (F1 < 0.75)*

	precision	recall	f1-score	support
dew	0.97	0.92	0.95	103
fogsmog	0.90	0.88	0.89	139
frost	0.78	0.68	0.72	68
glaze	0.73	0.83	0.78	106
hail	0.89	0.99	0.94	93
lightning	0.96	0.92	0.94	59
rain	0.74	0.91	0.82	68
rainbow	1.00	0.91	0.95	32
rime	0.88	0.88	0.88	182
sandstorm	0.86	0.89	0.87	88
snow	0.87	0.65	0.75	92
accuracy			0.86	1030
macro avg	0.87	0.86	0.86	1030
weighted avg	0.87	0.86	0.86	1030

The model correctly classifies 86% of all test images.

- *High precision (0.87 macro avg) means most predicted classes are correct*
- *High recall (0.86 macro avg) indicates the model successfully detects most true instances*
- *F1 Score:Balanced measure combining precision and recall (0.86 macro avg).*

OurInception_V1 Model

- *High performance: dew, hail, lightning, rainbow (F1 > 0.90)*
- *Lower performance: frost and snow (F1 < 0.75)*

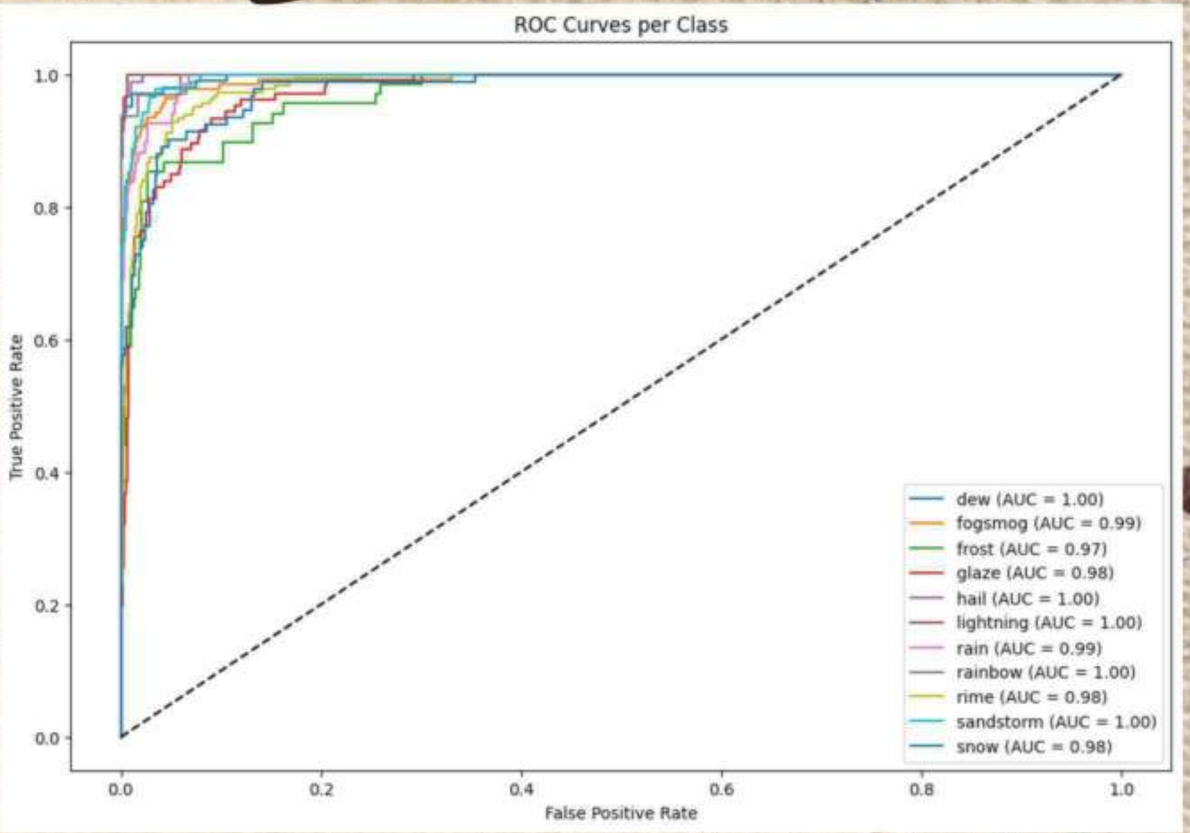
	precision	recall	f1-score	support
dew	0.97	0.92	0.95	103
fogsmog	0.90	0.88	0.89	139
frost	0.78	0.68	0.72	68
glaze	0.73	0.83	0.78	106
hail	0.89	0.99	0.94	93
lightning	0.96	0.92	0.94	59
rain	0.74	0.91	0.82	68
rainbow	1.00	0.91	0.95	32
rime	0.88	0.88	0.88	182
sandstorm	0.86	0.89	0.87	88
snow	0.87	0.65	0.75	92
accuracy			0.86	1030
macro avg	0.87	0.86	0.86	1030
weighted avg	0.87	0.86	0.86	1030

The model correctly classifies 86% of all test images.

- *High precision (0.87 macro avg) means most predicted classes are correct*
- *High recall (0.86 macro avg) indicates the model successfully detects most true instances*
- *F1 Score:Balanced measure combining precision and recall (0.86 macro avg).*

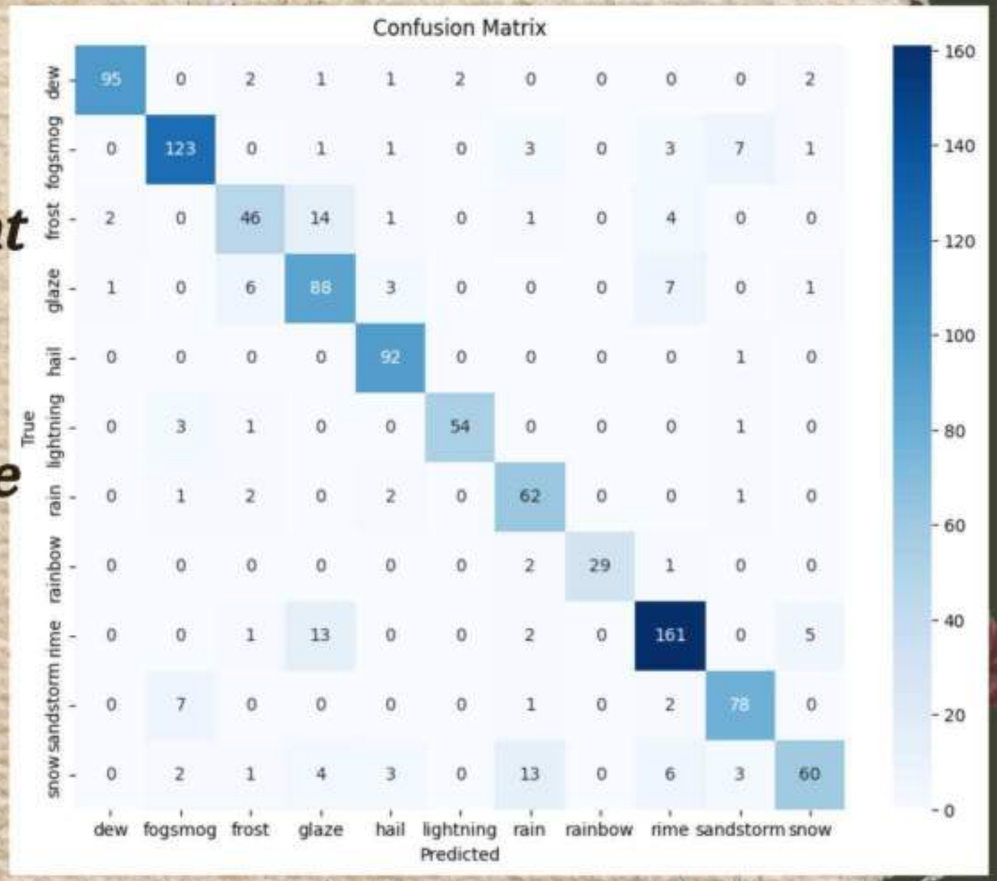
OurInception_V1 Model

- An AUC of 1.00 (e.g., dew, hail, lightning, rainbow, sandstorm) indicates a perfect classification for that class.
- Scores like 0.99, 0.98, and 0.97 indicate excellent classification performance, very close to perfect



OurInception_V1 Model

- 3. The "Rime" Class
- The Rime class appears to be the most accurately predicted class with 161 true positives. It also seems to have a high count of true instances (summing across the "rime" row).
- 4. Model's Weakness
- The most significant confusion seems to be between:
- Frost (True) ->Glaze (Predicted) (14 instances)
- Glaze (True) ->Rime (Predicted) (13 instances)
- Sandstorm (True) -> Snow (Predicted) (20 instances)



	ResNet50	MobileNet	VGG19	Inception v1
Pros	One of the main advantages is its ability to train very deep networks with hundreds of layers. ResNet-50 is its ability to achieve state-of-the-art results in a wide range of image-related tasks such as object detection, image classification, and image segmentation. Residual blocks allow for better generalization by reducing the risk of overlearning, which means that ResNet-50 is able to maintain high performance not only on training data but also on new data that it has not seen before. https://www.innovatiana.com/en/post/discover-resnet-50#what-are-the-advantages-of-the-resnet-50-architecture-compared-to-previous-models. Despite its depth, ResNet-50 (~25.6M parameters) outperforms VGG-19 (~143M parameters) with fewer resources, thanks to bottleneck designs that optimize computational efficiency.	Efficiency: MobileNet is designed specifically for mobile and embedded devices, making it highly efficient in terms of processing power and memory usage. Speed: The model is optimized for fast inference, making it ideal for real-time applications such as object detection and image recognition on mobile devices. Accuracy: Despite its focus on efficiency, MobileNet still achieves competitive accuracy in various vision tasks, making it suitable for a wide range of applications. https://www.iterate.ai/ai-glossary/mobilenet-for-mobile-embedded-vision-applications	• Simple and elegant architecture • Excellent feature extractor (used in transfer learning) • High accuracy on CIFAR and ImageNet https://ai.plainenglish.io/reproducing-vgg19-from-scratch-on-cifar-10-using-pytorch-a-deep-dive-into-very-deep-cnns-e6e8b0b8573c	Computational Efficiency: Achieved high accuracy (ImageNet winner) with significantly fewer parameters and operations than predecessors by using 1x1 convolutions to reduce depth before expensive 3x3 and 5x5 convolutions. Multi-Scale Feature Extraction: The Inception module processes features at different scales (1x1, 3x3, 5x5 convolutions, max pooling) in parallel, capturing rich information, notes and Vanishing Gradient Mitigation: Used auxiliary classifiers (side branches) during training, which helped gradients flow better in deep layers and improved convergence. https://medium.com/data-science/deep-learning-understand-the-inception-module-56146866e652
Cons	• Overfitting on Small Datasets • Shortcut connections mitigate vanishing gradients by providing a direct path for gradient propagation. This eliminates degradation, as evidenced by ResNet-152's 3.57% top-5 error rate on ImageNet, compared to VGG-19's 6.7%, despite being far deeper (152 vs. 19 layers) • https://medium.com/@deblinab101/deep-dive-into-residual-networks-a-thorough-review-of-resnet-architectures-44e963f3a53a	Complex tasks: MobileNet may not be as suitable for complex vision tasks that require a higher level of precision and detailed feature extraction. Trade-off between speed and accuracy: In some cases, the speed optimization of MobileNet may come at the cost of some degree of accuracy, particularly in challenging vision scenarios. https://www.iterate.ai/ai-glossary/mobilenet-for-mobile-embedded-vision-applications	Heavy: ~140 million parameters High memory & compute requirements Lacks architectural innovations like residuals or attention https://ai.plainenglish.io/reproducing-vgg19-from-scratch-on-cifar-10-using-pytorch-a-deep-dive-into-very-deep-cnns-e6e8b0b8573c	Design Complexity: The module itself was intricate, requiring careful tuning of filter sizes and connections, note and Medium. • Information Loss: The 5x5 convolutions could still lead to information loss compared to smaller kernels, which was addressed in later versions, according to Medium.Overfitting Potential (Initial): Like many deep models, it could overfit, though techniques like dropout (especially on auxiliary branches) helped, https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45169.pdf

	ResNet50 At epoch 14	MobileNet At epoch 31	VGG19 At epoch 5	Inception v1 At epoch 5
ACC_train	0.9951	0.9903	0.6297	0.845
Acc_val	0.8786	0.8386	0.6029	0.861
LOSS_train	0.0596	0.0629	1.0675	0.459
Loss_val	0.4852	0.4978	1.2315	0.410

ResNet50 achieved the highest training accuracy (99.51%) with a very low training loss (0.0596), indicating strong learning capability. However, the validation accuracy (87.86%) is noticeably lower than the training accuracy, and the validation loss increases to 0.4852. This gap suggests overfitting.

Although MobileNet required a higher number of epochs (31) to reach this performance, it provides a good rade-off between accuracy and computational efficiency. However, its validation performance remains lower than ResNet50 and Inception v1.

VGG19 showed the weakest performance among all evaluated models.

Inception v1 achieved the best overall performance in terms of generalization. It recorded the highest validation accuracy (86.1%) and the lowest validation loss (0.410), while maintaining a relatively small gap between training and validation metrics.

	ResNet50	MobileNet	VGG19	Inception v1
ACC_Test	0.84	0.827	0.569	0.86
precision	0.85	0.84	0.567	0.87
Recall	0.83	0.829	0.543	0.86
F1 Score	0.84	0.83	0.543	0.86

ResNet50 achieved a strong test performance:These results indicate a well-balanced classifier with good precision–recall trade-off. However, its performance remains slightly lower than Inception v1.

MobileNet
The results demonstrate consistent performance across all metrics, making MobileNet a reliable and efficient model.

VGG19 exhibited the weakest performance on the test dataset,

Inception v1 achieved the best test performance across all evaluation metrics, recording an accuracy of 86%, precision of 87%, recall of 86%, and an F1-score of 86%

- *Best Overall Test Model: Inception v1*

  High accuracy across all categories
  Low validation loss
  Balanced F1 score of 0.86

- *ResNet50: Excellent in some categories, but suffers from overfitting and slight dispersion between Frost and Glaze.*

- *MobileNet: Good in specific categories, but average overall performance.*

- *VGG19: Very weak → possibly unsuitable for datasets or requires retraining and fine-tuning.*

Thank You