Deep Learning

PROJECT

Team: 10:40

Contents

Dataset And Prepressing

Dataset

Classes

NUM: 5 classes

- cloudy 6702
- snowy 1875
- sunny 6274
- foggy 1261
- rainy 1927

Has High Imbalance between the classes

Solution : Data Augmentation

After Augmentation

- cloudy 6702
- snowy 5625
- sunny 6274
- foggy 5044
- rainy 5781

Preprocessing

Common steps

1. Images are loaded from class-specific folders.
2. Images are resized to a consistent size for the model:
   - VGG19: 128×128
   - Transfer models: 224×224
3. Class labels are automatically encoded as integers.
4. Pixel values are normalized, either by simple scaling or model-specific preprocessing.
5. Dataset is split into train, validation, and sometimes test sets.
6. Data augmentation is applied during training to improve generalization: Flips, rotations, zooms, contrast adjustments are common.

Preprocessing

Different steps

VGG19 (trained from scratch)

Framework: TensorFlow / Keras
Input size: 128×128
Normalization: divide by 255 → [0,1]
Augmentation is **offline**, saved to disk, and mostly applied to **minority classes** (rainy, snowy, foggy).

Inception V1 (transfer learning)

Framework: PyTorch
Uses **ImageNet pre-trained weights**.
Input size: 224×224
Normalization: ImageNet mean/std
Augmentation is **on-the-fly** (flip, rotation, zoom, contrast) for all training images.

Preprocessing

Different steps

MobileNetV2 (transfer learning)

Framework: TensorFlow / Keras

Uses **ImageNet pre-trained weights**.

Input size: 224×224

Normalization: MobileNetV2-specific preprocessing → [-1, 1]

On-the-fly augmentation: flips, stronger rotations, zoom, contrast

Corrupted images are detected and removed before training.

ResNet50 (transfer learning)

Framework: PyTorch

Uses **ImageNet pre-trained weights**.

Input size: 224×224

Normalization: ImageNet mean/std

On-the-fly augmentation: flips, rotations, color jitter, random resized crop

Models

VGG-19

VGG → Visual Geometry Group
19 → total number of **layers with learnable parameters**
16 convolutional layers
3 fully connected layers

Architecture Summary

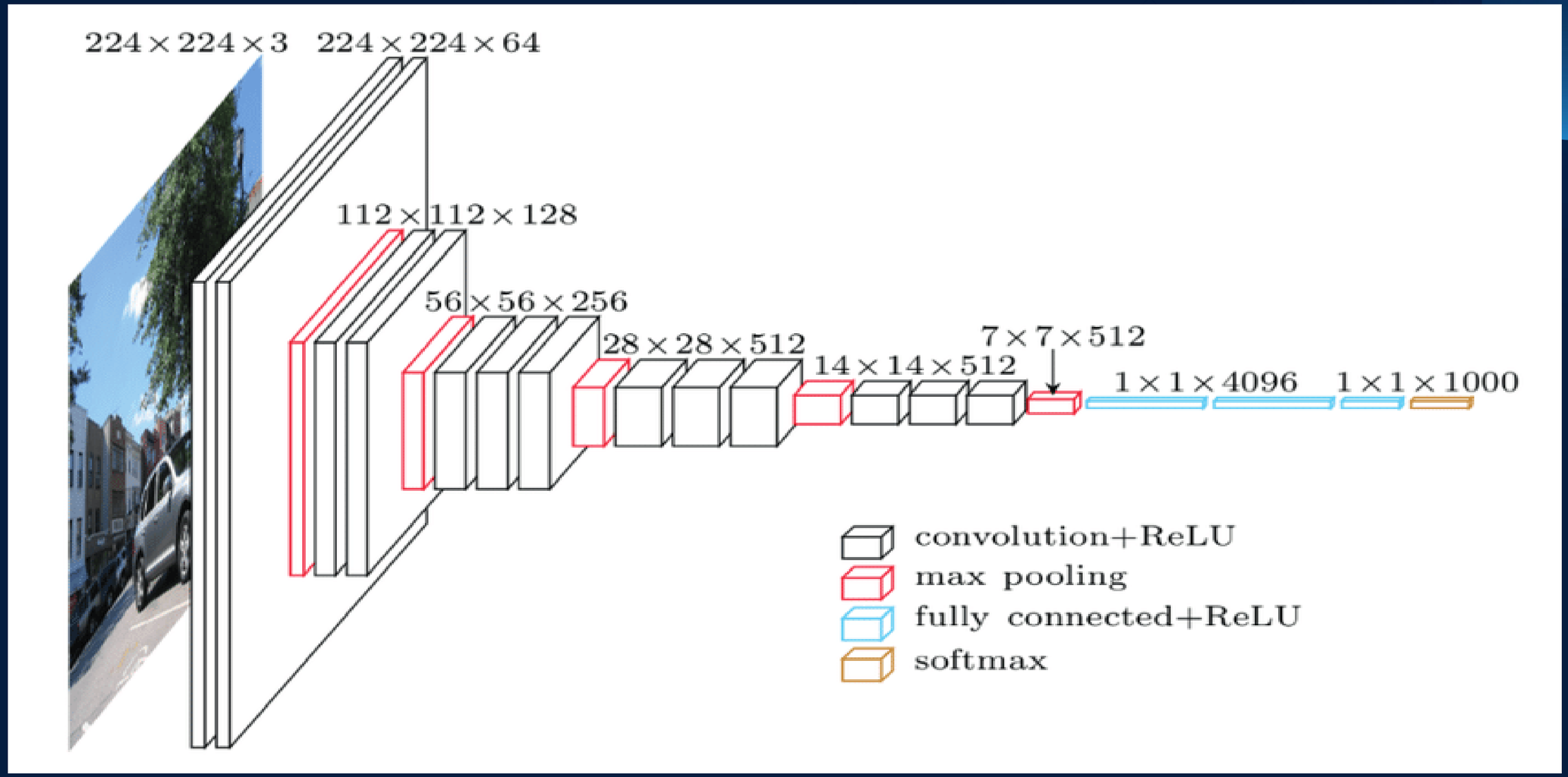
- Uses **small 3×3 convolution filters**
- Uses **max pooling (2×2)** to reduce spatial size
- Uses **ReLU** activation
- Ends with **fully connected layers + Softmax**

Input Requirements

- **Image size: 224 × 224 × 3**
- **Images must be resized and normalized**

Number of Parameters:

- **About 143 million parameters**
- **Very large and memory-heavy**



Paper: https://www.researchgate.net/publication/346716209_A_Convolutional_Neural_Network_Classifier_VGG-19_Architecture_for_Lesion_Detection_and_Grading_in_Diabetic_Retinopathy_Based_on_Deep_Learning

ResNet

ResNet → Residual Network
50 → total number of layers

Architecture Summary

Uses 3×3 convolution filters (and 1×1 convolutions for dimension matching)

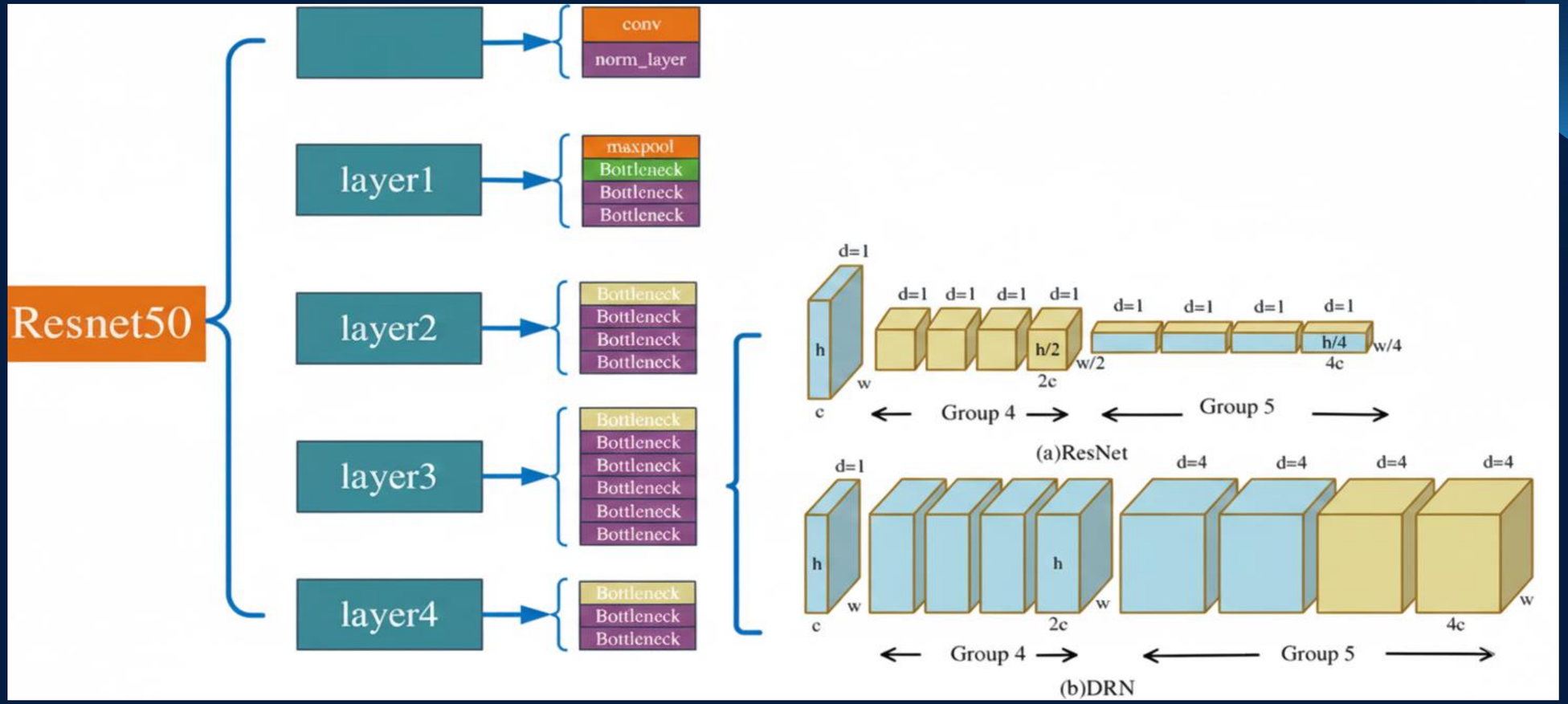
- Uses stride-based downsampling instead of heavy max pooling
- Uses Batch Normalization + ReLU activation
- Uses residual (skip) connections to add input directly to output
- Ends with Global Average Pooling + Fully Connected layer + Softmax

Input Requirements

- Image size: 224 × 224 × 3
- Images must be resized and normalized

Number of Parameters:

- About 25.6 million parameters



Paper:

- https://arxiv.org/abs/1512.03385
- https://pytorch.org/vision/stable/models/resnet.html
- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Inception V1

Inception v1 → GoogLeNet
22 → total number of layers

Architecture Summary

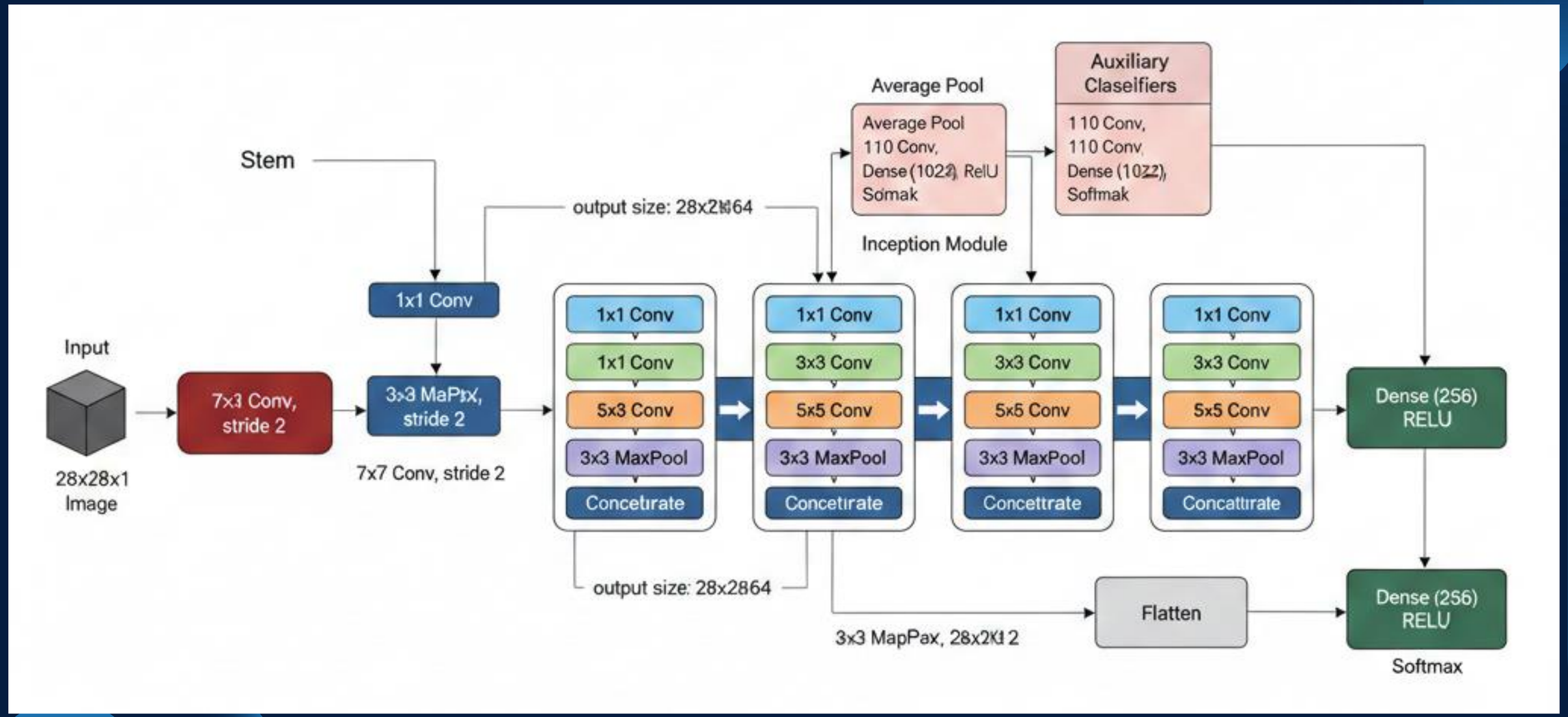
- Uses Inception modules (parallel convolutions)
- Combines 1×1, 3×3, and 5×5 filters in the same layer
- Uses 1×1 convolutions for dimensionality reduction
- Reduces computational cost while increasing network depth
- Addresses overfitting using auxiliary classifiers
- Allows building deep and efficient networks

Input Requirements

- Image size: 224 × 224 × 3
- Images must be resized and normalized (ImageNet preprocessing)

Number of Parameters

- About 6–7 million parameters



Paper: https://www.researchgate.net/figure/Architecture-of-inception-and-inception-V1_fig20_379189553

MobileNet

MobileNet → Mobile Network
28 → total number of layers

Architecture Summary

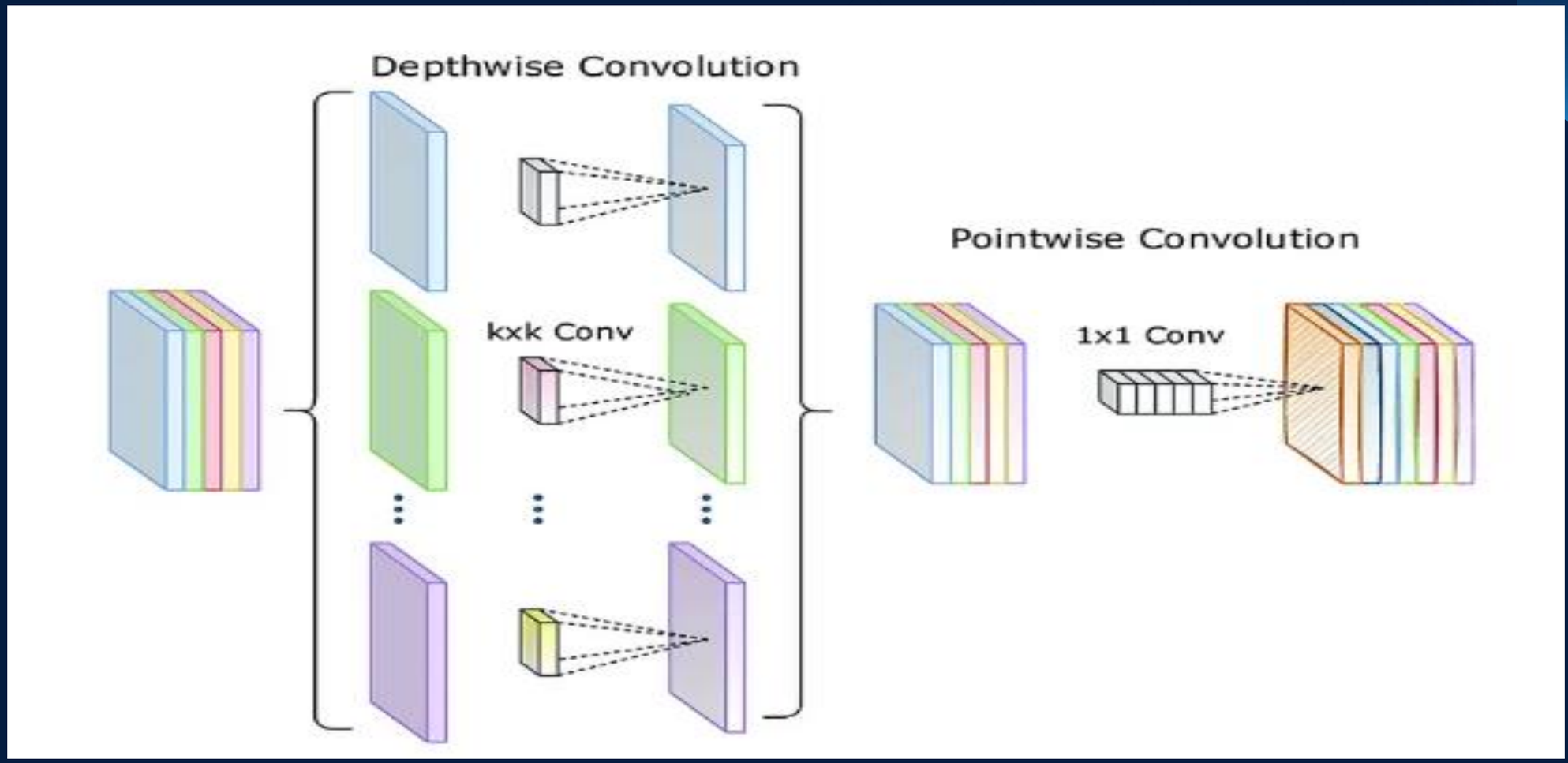
- Uses depthwise separable convolutions
- Splits standard convolution into:
- Depthwise convolution (per channel)
- Pointwise 1×1 convolution (channel mixing)
- Greatly reduces computation and number of parameters
- Uses Batch Normalization + ReLU (ReLU6) activation
- Uses stride-based downsampling instead of max pooling
- Ends with Global Average Pooling + Fully Connected layer + Softmax

Input Requirements

- **Image size: 224 × 224 × 3**
- **Images must be resized and normalized (ImageNet preprocessing)**

Number of Parameters

- **About 4.2 million parameters**
- **Very small and lightweight compared to VGG, ResNet, and Inception**



Paper:

- https://viso.ai/deep-learning/mobilenet-efficient-deep-learning-for-mobile-vision/
- https://arxiv.org/abs/1704.04861

Comparisons

Comparisons on dataset

VGG-19

✓ Pros

- Strong at learning low-level features (edges, textures)
- Works reasonably well for clear classes (sunny vs snowy)
- Easy to fine-tune using transfer learning
- Good baseline to compare other models

✗ Cons

- Overfitting risk due to many parameters
- Needs heavy augmentation + dropout
- Biased toward majority classes (sunny, cloudy)
- Slow training on ~18K images

Inception V1

- ✓ Pros
- Captures multi-scale features → great for weather textures
- Fewer parameters → less overfitting
- Better discrimination between foggy / cloudy / rainy
- Efficient on medium-size datasets

- ✗ Cons
- Still affected by class imbalance
- Harder to tune
- Slightly outdated compared to newer Inception versions

Comparisons on dataset

ResNet

✓ Pros

- Excellent feature depth for subtle differences
- Strong performance on foggy vs cloudy
- Handles imbalance better when combined with:Class weights
- Data augmentation
- Stable training

✗ Cons

- More computation than Inception/MobileNet
- Risk of learning bias without class weighting
- Overkill if not tuned properly

MobileNet

- ✓ Pros
- Fast training and inferenceLow risk of overfitting
- Good baseline for deployment
- Handles imbalance better than VGG (fewer parameters)

- ✗ Cons
- Struggles with fine-grained differences
- Lower accuracy for foggy & rainy
- May confuse cloudy vs foggy

Compare Results

Comparisons on Accuracy

Model	Validation Accuracy
MobileNetV2	85.26%
Inception V1	85.34%
VGG19 (scratch)	70.88%
ResNet50	81.48%

Comparisons on precision, Recall , F1

Inception V1

Classification Report:

	precision	recall	f1-score	support
cloudy	0.86	0.79	0.82	1341
foggy	0.79	0.92	0.85	252
rainy	0.82	0.79	0.80	385
snowy	0.82	0.88	0.85	375
sunny	0.84	0.88	0.86	1255
accuracy			0.84	3608
macro avg	0.82	0.85	0.84	3608
weighted avg	0.84	0.84	0.84	3608

MobileNet V2

--- CLASSIFICATION REPORT ---

	precision	recall	f1-score	support
cloudy	0.87	0.81	0.84	1357
foggy	0.91	0.88	0.90	252
rainy	0.84	0.82	0.83	371
snowy	0.79	0.93	0.85	374
sunny	0.85	0.88	0.87	1241
accuracy			0.85	3595
macro avg	0.85	0.86	0.86	3595
weighted avg	0.85	0.85	0.85	3595

ResNet50

	precision	recall	f1-score	support
cloudy	0.73	0.91	0.81	967
foggy	0.74	0.87	0.80	190
rainy	0.89	0.61	0.72	273
snowy	0.85	0.81	0.83	294
sunny	0.92	0.77	0.84	983
accuracy			0.81	2707
macro avg	0.83	0.79	0.80	2707
weighted avg	0.83	0.81	0.81	2707

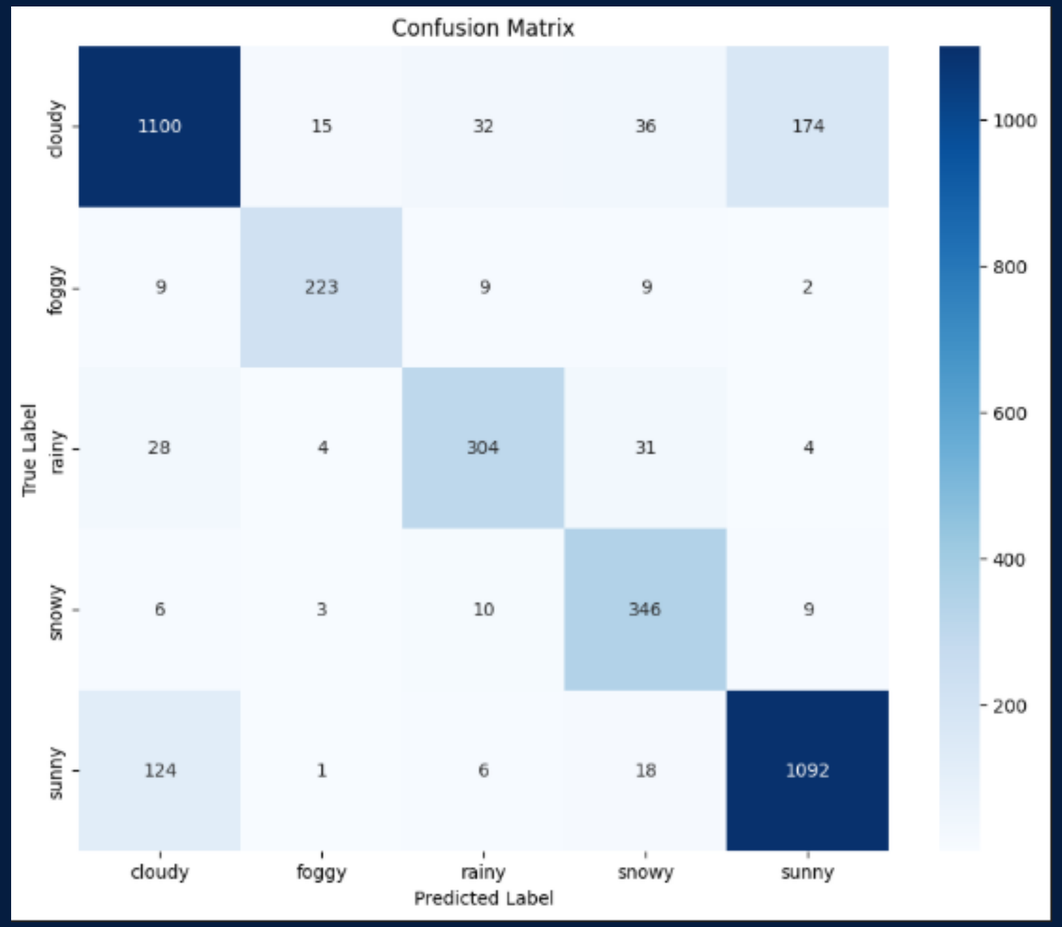
VGG-19

Classification Report:

	precision	recall	f1-score	support
cloudy	0.67	0.48	0.56	158
snowy	0.70	0.76	0.73	159
sunny	0.77	0.81	0.79	156
foggy	0.84	0.86	0.85	157
rainy	0.57	0.64	0.61	170
accuracy			0.71	800
macro avg	0.71	0.71	0.71	800
weighted avg	0.71	0.71	0.71	800

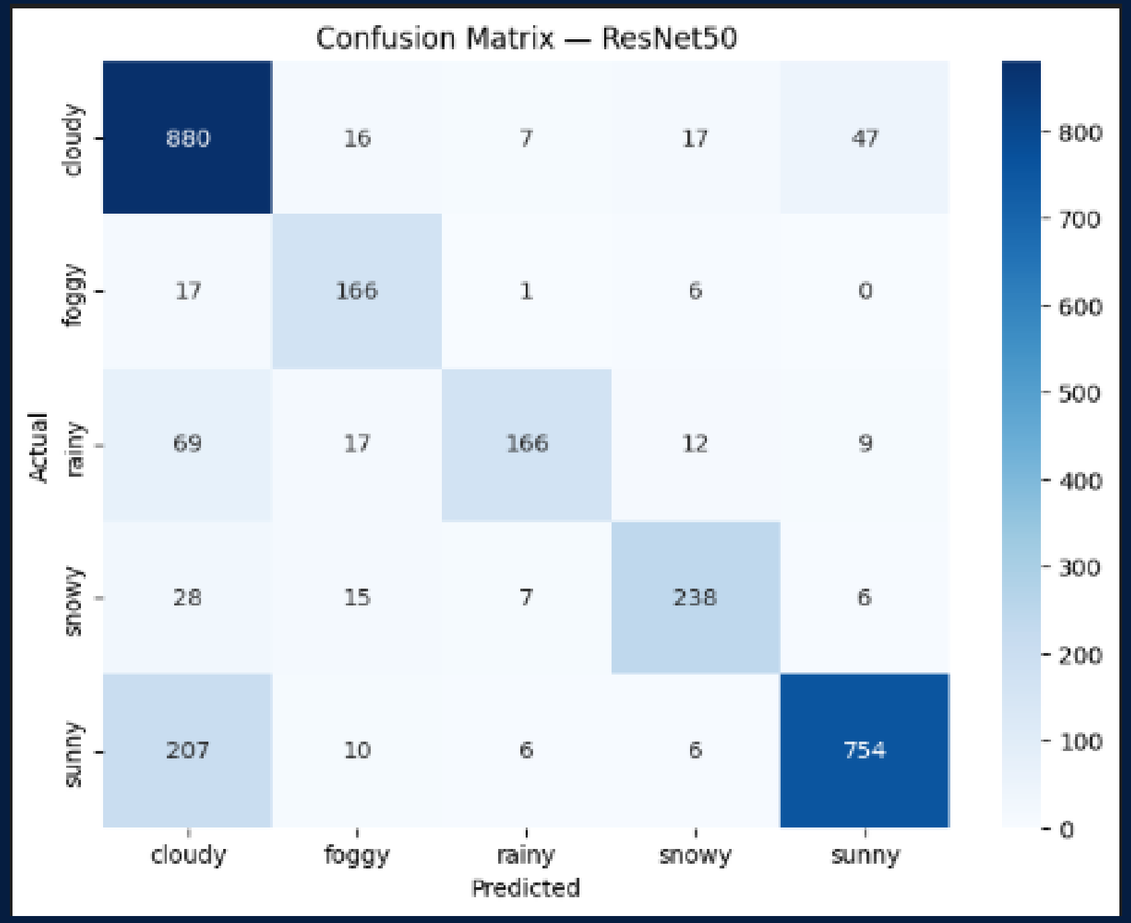
Comparisons on Confusion Matrix

Inception V1

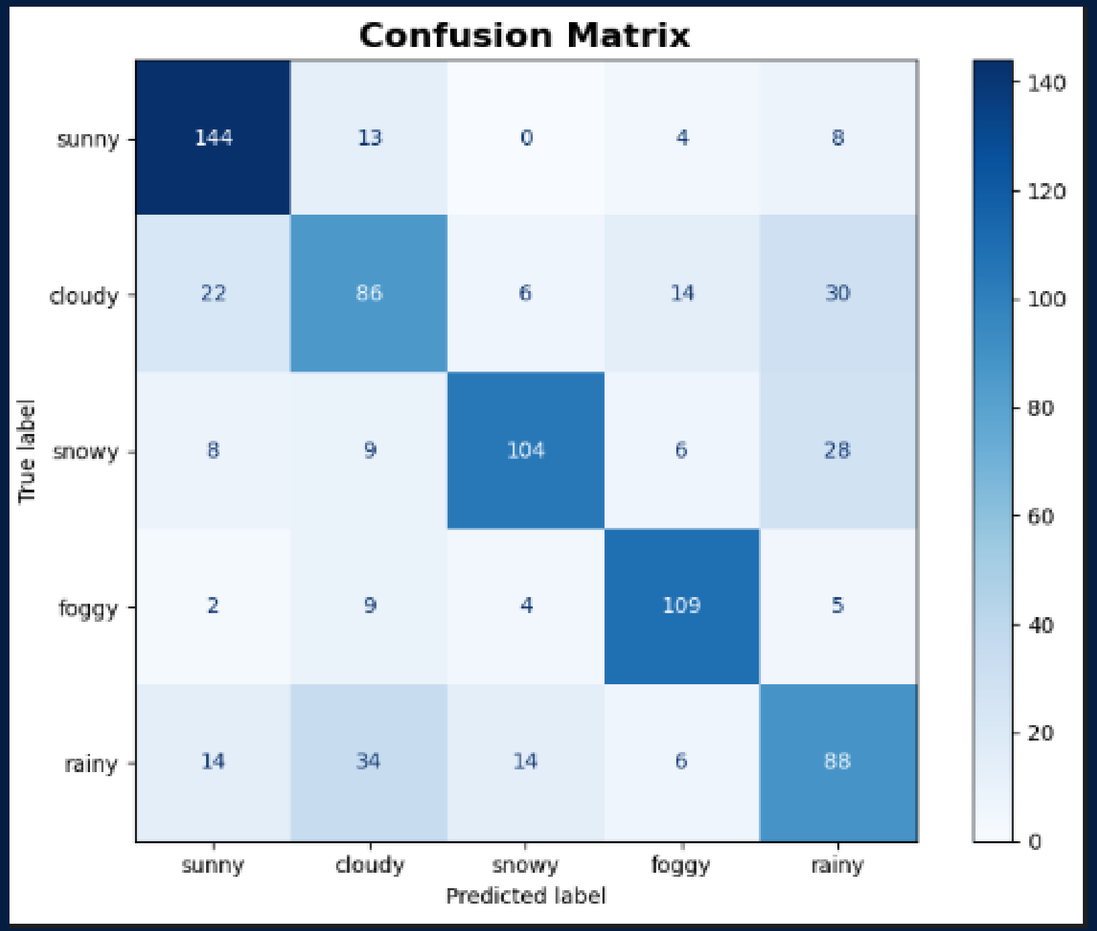


MobileNet V2

Comparisons on Confusion Matrix

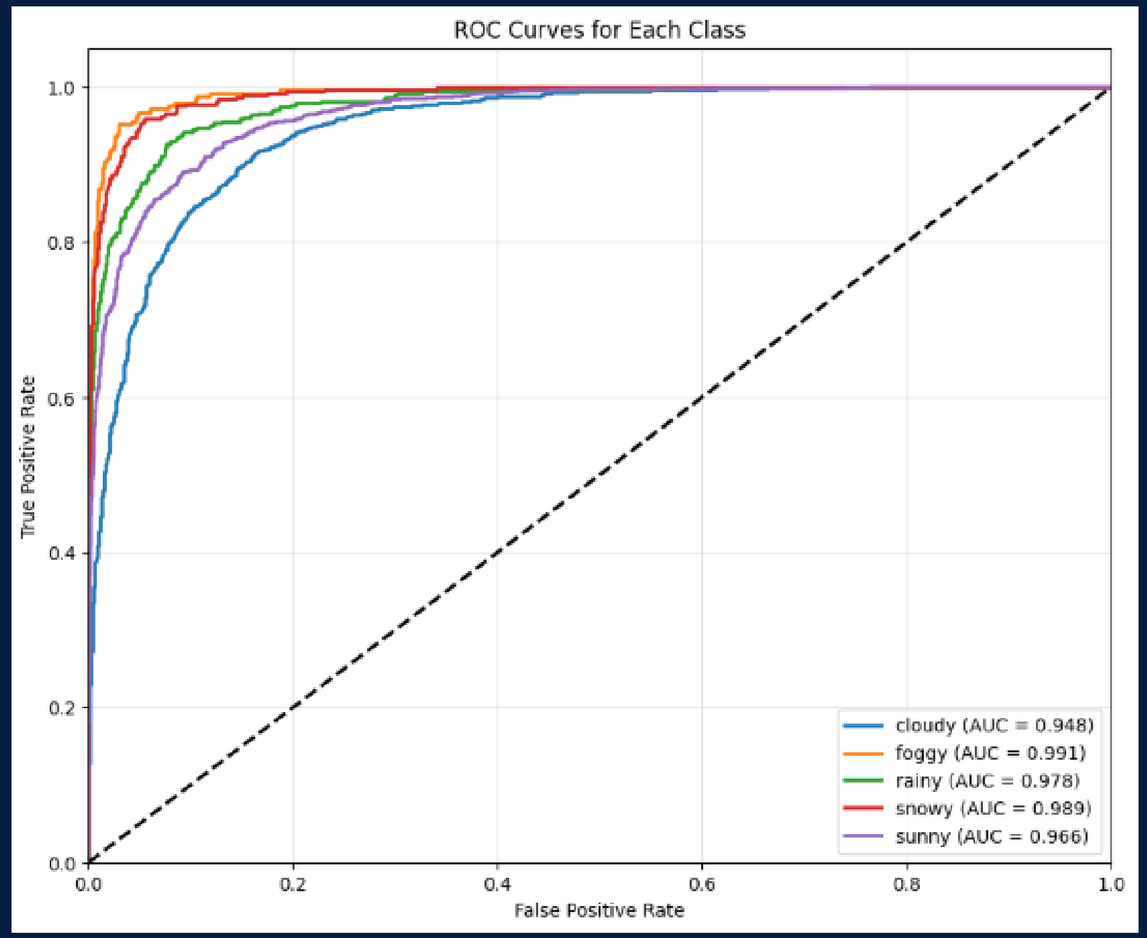


ResNet50

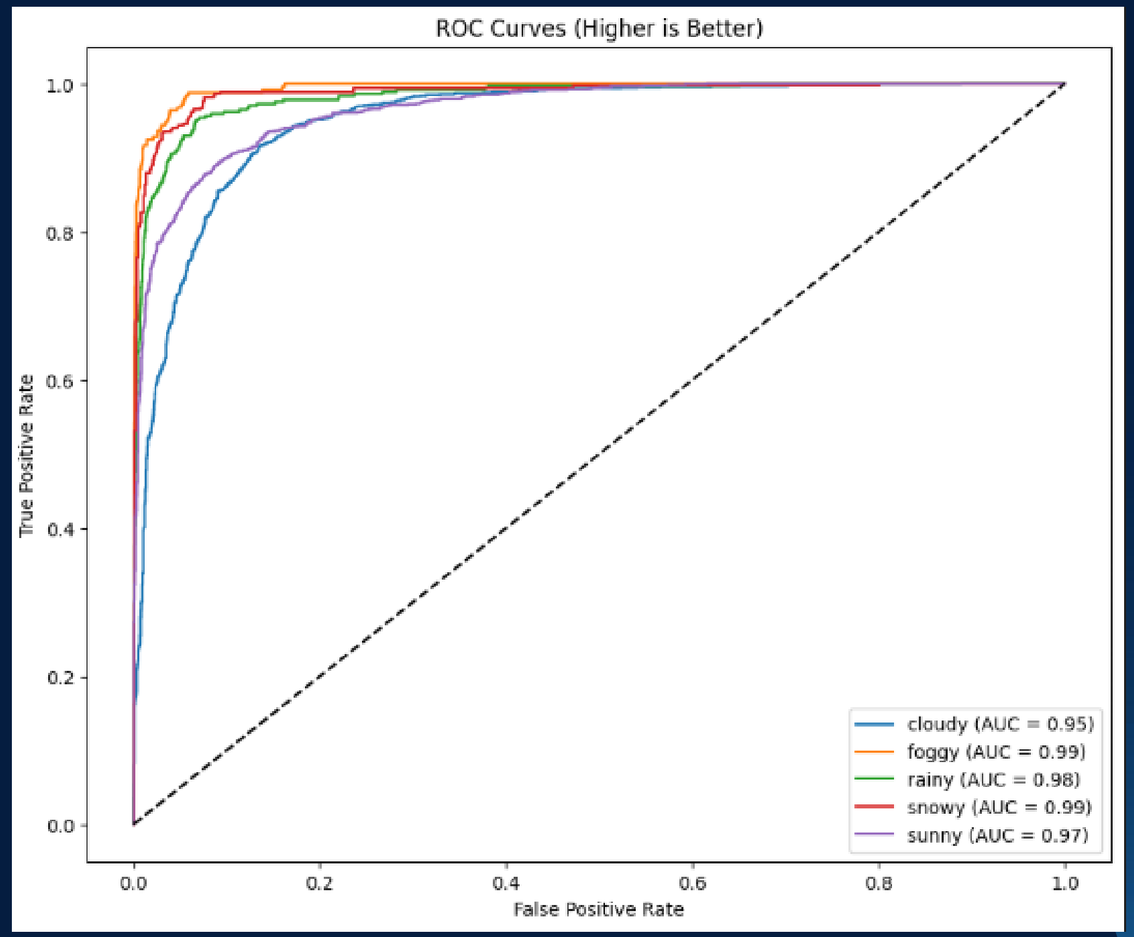


VGG-19

Comparisons on ROC & AUC

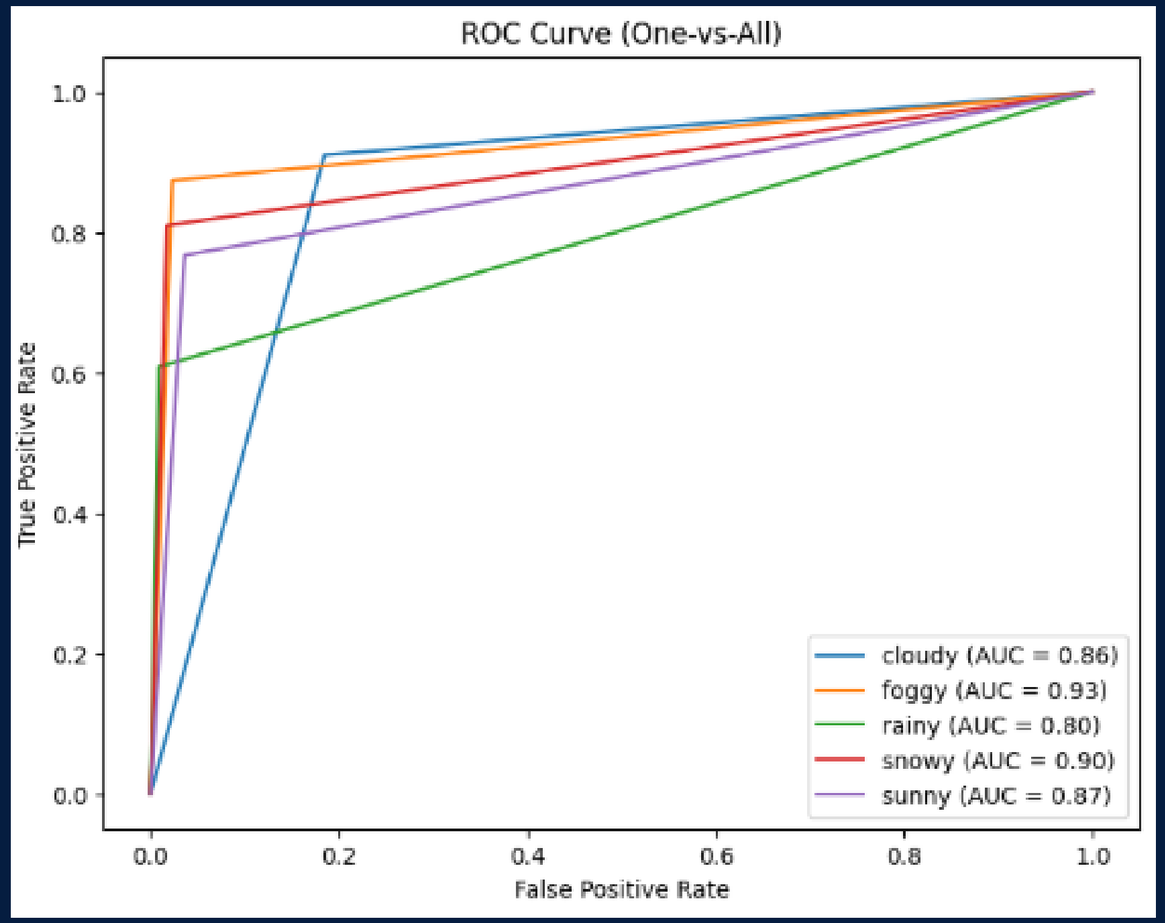


Inception V1

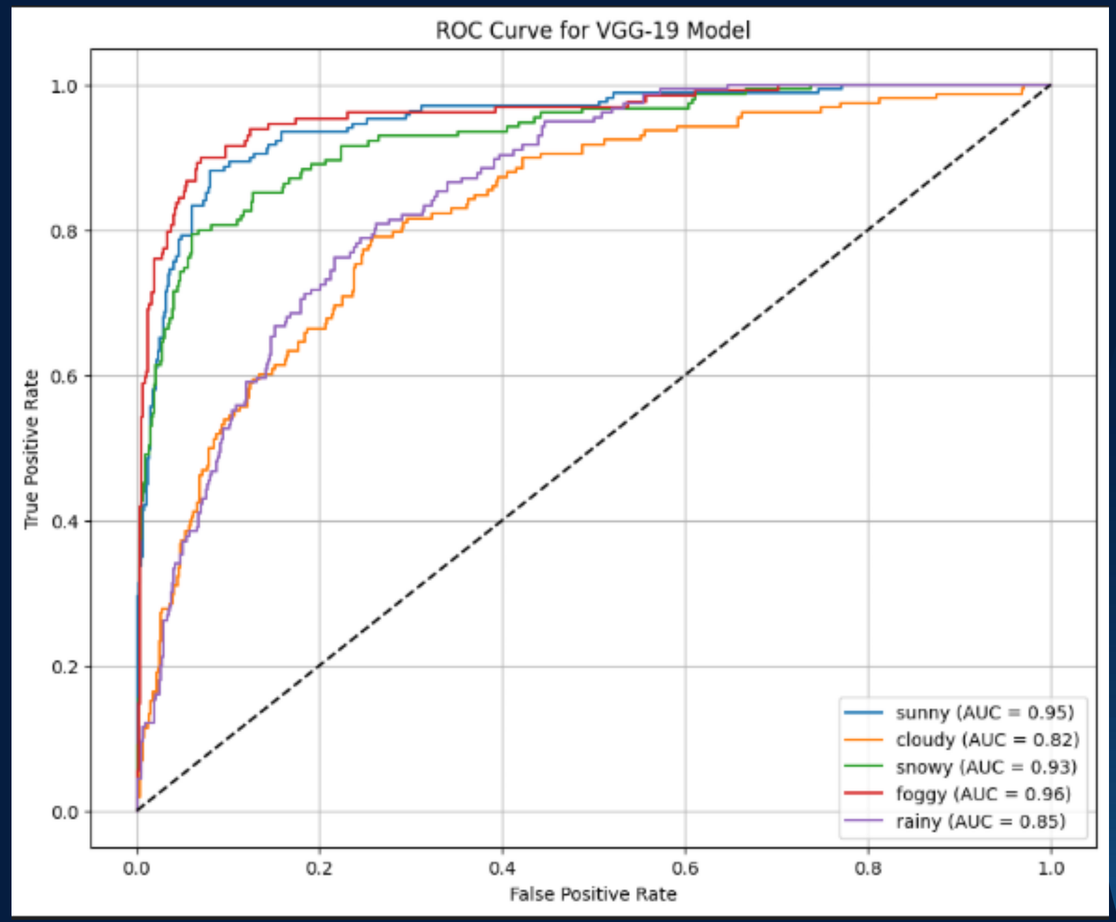


MobileNet V2

Comparisons on ROC & AUC



ResNet50



VGG-19

GitHub

README

Weather Image Classification with Deep Learning

This project implements and compares various Deep Learning architectures (ResNet, VGG19, InceptionV1, MobileNet) to classify weather conditions from images. Using transfer learning, the models are trained to distinguish between five different weather types: Cloudy, Foggy, Rainy, Snowy, and Sunny.

📌 Project Overview

Goal: Build an accurate image classifier for weather conditions.
Dataset: 5-Class Weather Status Image Classification from Kaggle

Dataset Details

The dataset consists of images categorized into 5 classes:

- Cloudy
- Foggy
- Rainy
- Snowy
- Sunny

Pre-processing includes resizing (224x224), normalization, and da

Deep_Learning_Project Public

Pin | Watch 0

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

Reham-Ahmed19 Update README.md — 57615ec · 2 hours ago — 9 Commits

Inception-V1-Model.ipynb	Add files via upload	now
MobileNet-V2-Model.ipynb	Add files via upload	now
README.md	Update README.md	2 hours ago
ResNet-50-Model.ipynb	Add files via upload	now
VGG-19-Model.ipynb	Add files via upload	now

README

THANK YOU