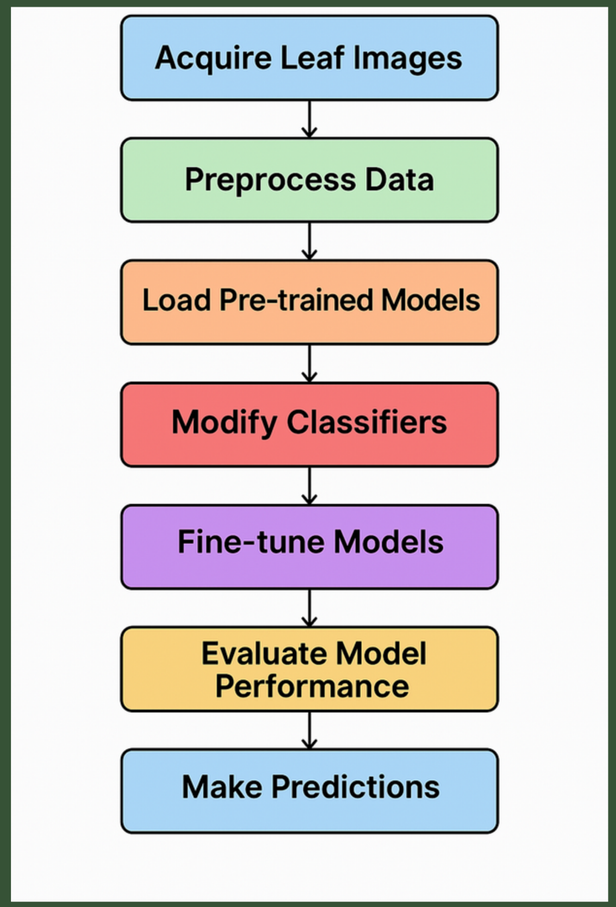
Comparative Analysis of Deep Learning Architectures for Image Classification

Presented by:
Hardik Chauhan

Date Presented:
April 7th, 2025

Introduction



- **Problem Statement** This study compares VGG, MobileNet, and ResNet architectures on the MepcoTropicLeaf dataset, evaluating performance, efficiency, and the impact of data augmentation to guide optimal model selection for real-world applications for image classification.
- **Objective**: The goal of this project is to implement and evaluate three deep learning models: VGG16, ResNet18, and MobileNetV2 to classify leaves in the MepcoTropicLeaf dataset. By fine-tuning pre-trained models, we aim to achieve high classification accuracy.

Dataset Overview

Dataset: MepcoTropicLeaf

- A plant species classification dataset consisting of **50 classes** of leaves.
- Total of **3777 images**, each representing a unique leaf sample for classification.
- Some examples of plant species included in the dataset are:
  - Indian Jujube (Ber)
  - Holy Basil (Tulsi)
  - Madagascar Periwinkle (Vinca)
- The dataset is designed for training and evaluating machine learning models, focusing on plant species identification through leaf images.
- **Source:** The dataset was obtained from **Kaggle**.

Models

VGG16

- Consists of 13 convolutional layers and 3 fully connected layers
- Uses 3x3 convolutions with ReLU activation
- MaxPooling layers after every few conv layers
- Final classifier has:
  - Two FC layers (4096 units each) + ReLU + Dropout
  - Output layer: 50 classes (Softmax)
- ~138 million parameters → High computational cost

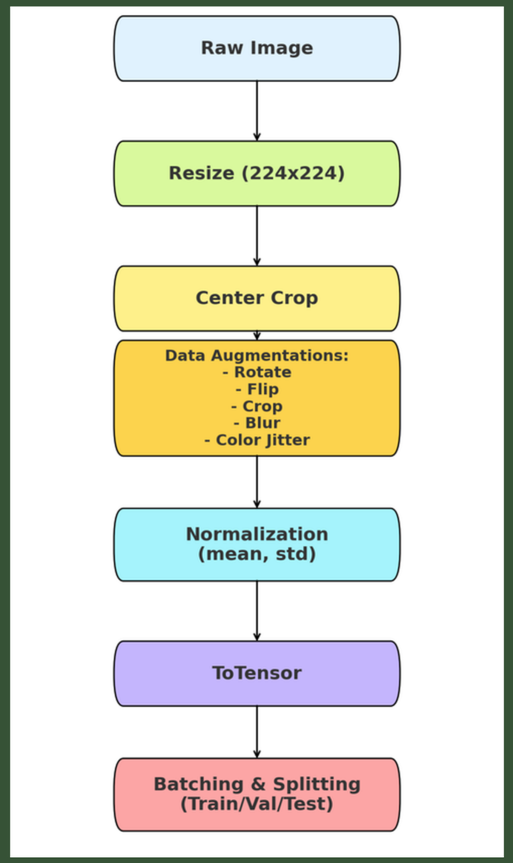
MobileNetV2

- Lightweight architecture optimized for mobile/edge devices
- Uses depthwise separable convolutions to reduce parameters
- Contains 17 residual blocks with inverted bottlenecks
- Modified classifier:
  - Hidden layer with 512 units + ReLU
  - Dropout (0.3) + Output: 50 classes
- ~3.4 million parameters → Highly efficient

ResNet18

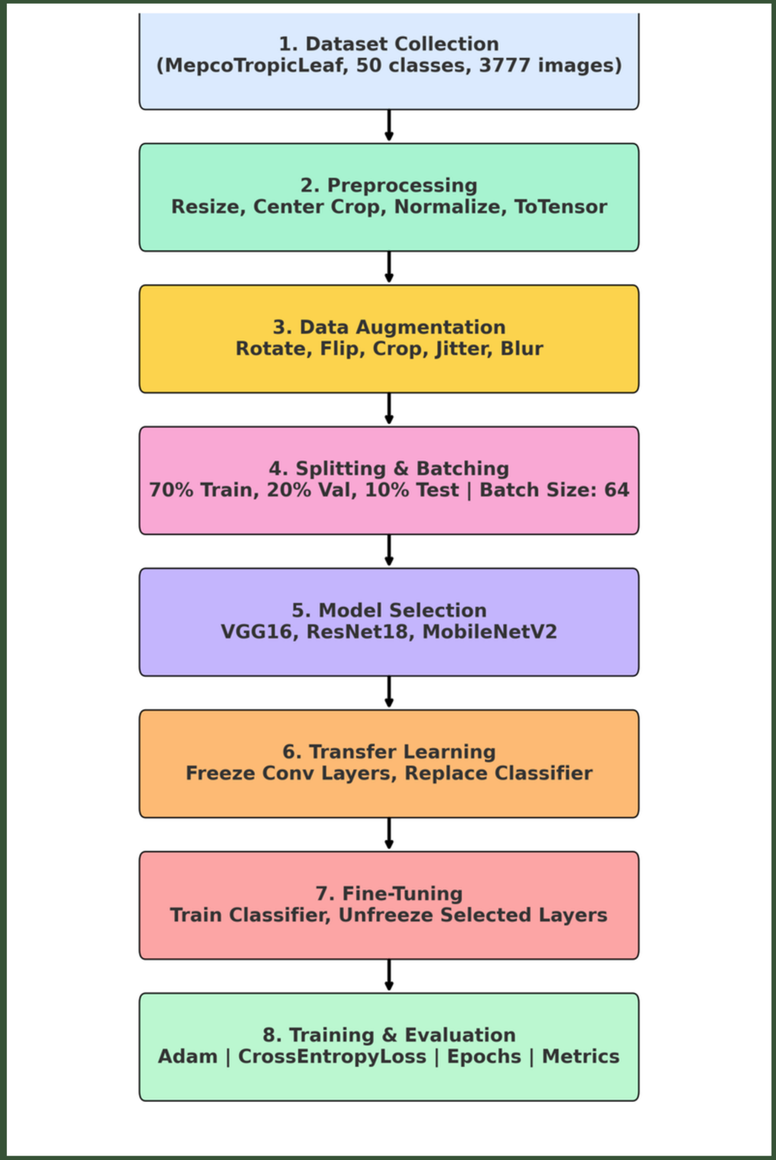
- Known for residual (skip) connections that solve vanishing gradient issues
- 17 convolutional layers grouped into 2 main residual blocks
- Custom classifier: Fully connected layers → 50 output classes
- ~11 million parameters → Balance of depth and efficiency

Data Preprocessing and Augmentation



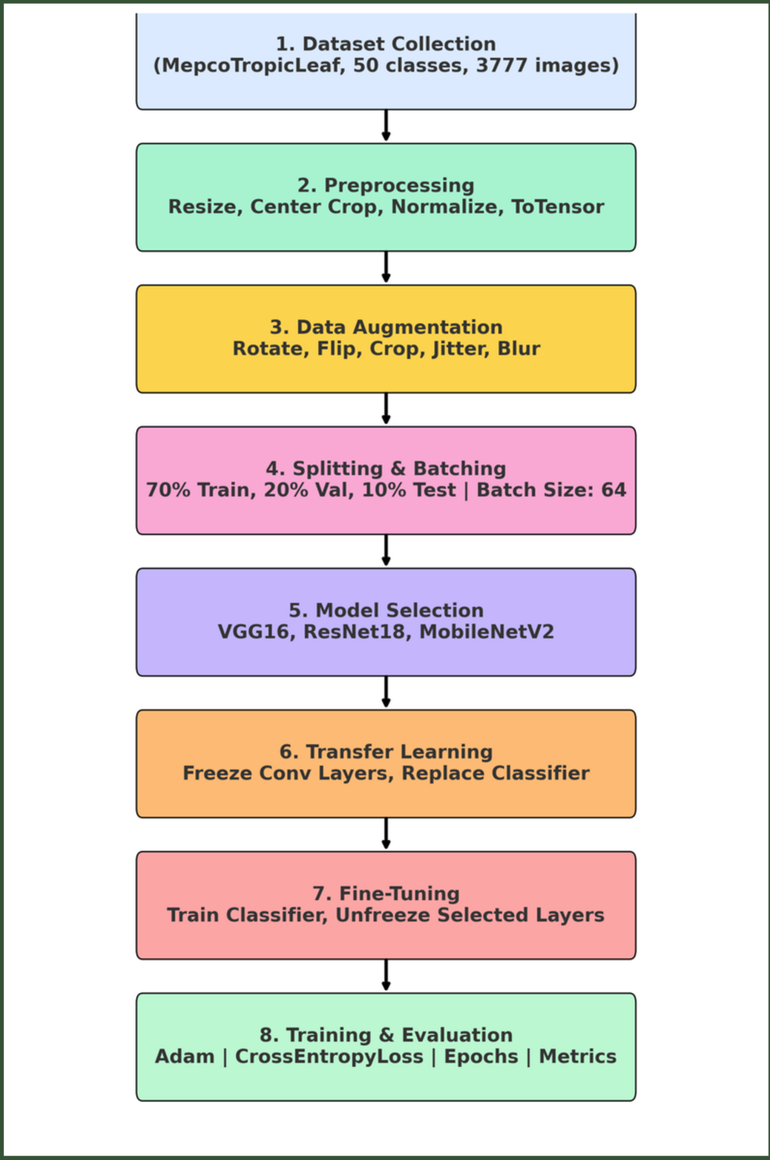
1. Data Augmentation Techniques
   - Random Rotation: ±30° → Helps model learn from different orientations
   - Horizontal Flip: 50% chance → Adds symmetry-based robustness
   - Color Jitter: Brightness, Contrast, Saturation, Hue
   - → Makes model less sensitive to lighting/color
   - Gaussian Blur: 20% chance
   - → Simulates camera blur or focus issues
2. Preprocessing Steps
   - Resize: All images resized to 224×224
   - → Matches input size for VGG16, ResNet18, MobileNetV2
   - Center Crop: Focuses on the main subject
   - Normalization:
   - → mean = [0.485, 0.456, 0.406]
   - → std = [0.229, 0.224, 0.225]
     - Matches ImageNet stats for pretrained models
   - ToTensor: Converts images to PyTorch tensors (scaled 0–1)

Methodology



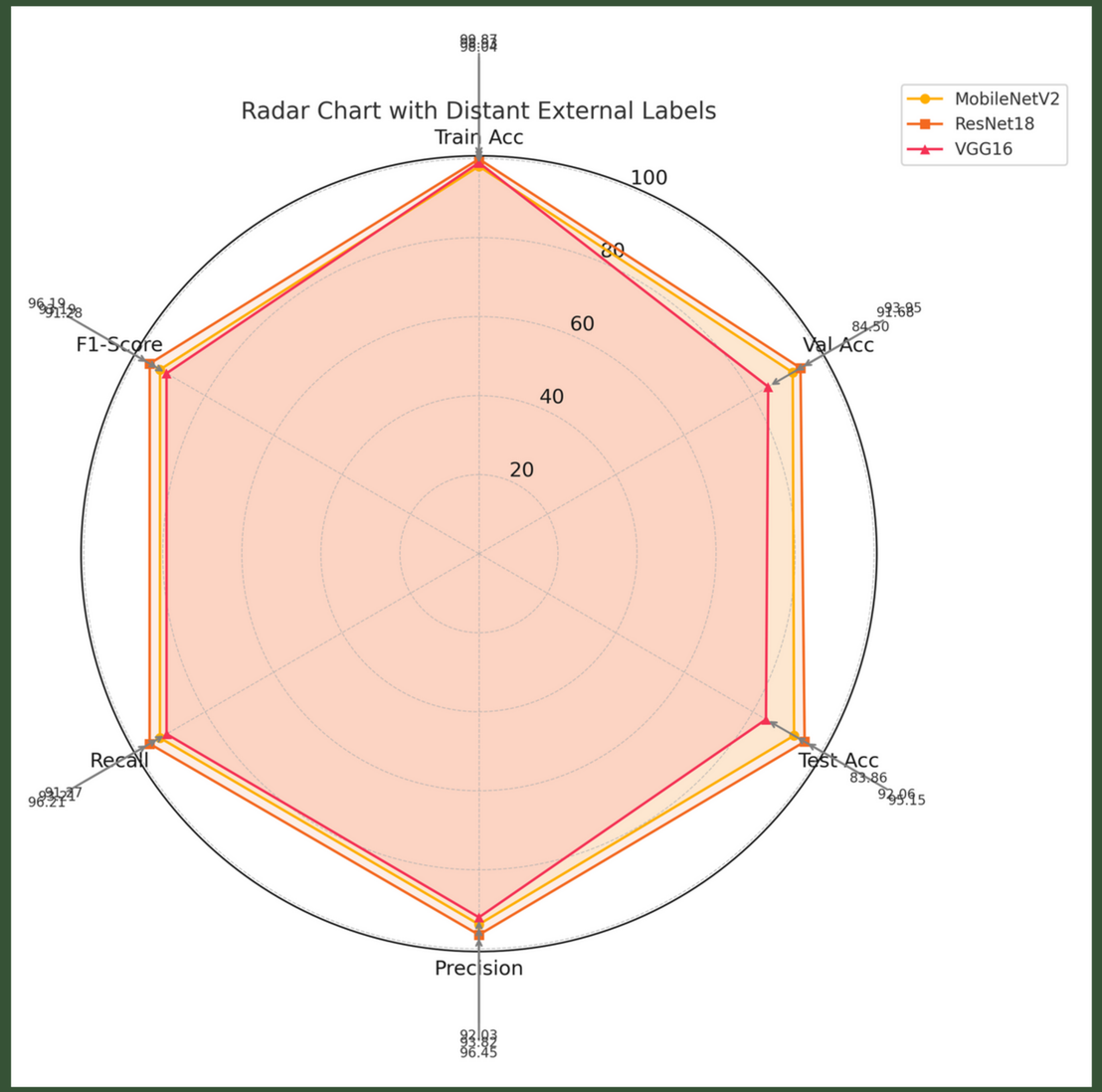
1. Dataset Collection
   - Dataset: MepcoTropicLeaf (from Kaggle)
   - Classes: 50 different leaf types
   - Images: 3,777 diverse samples
2. Preprocessing
   - Resizing: All images scaled to 224×224 pixels
   - Center Cropping: Focus on primary region of interest
   - Normalization: (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225])
   - ToTensor: Conversion to PyTorch tensors for model input (i.e. to [0,1])
3. Data Augmentation
   - Random Rotation (±30°)
   - Horizontal Flip
   - Resized Crop (80–100%)
   - Color Jitter (brightness, contrast, hue)
   - Gaussian Blur (20% chance)
4. Splitting & Batching
   - Dataset split into:
     - 70% Training
     - 20% Validation
     - 10% Testing
   - Batch size: 64 for efficient GPU usage

Methodology



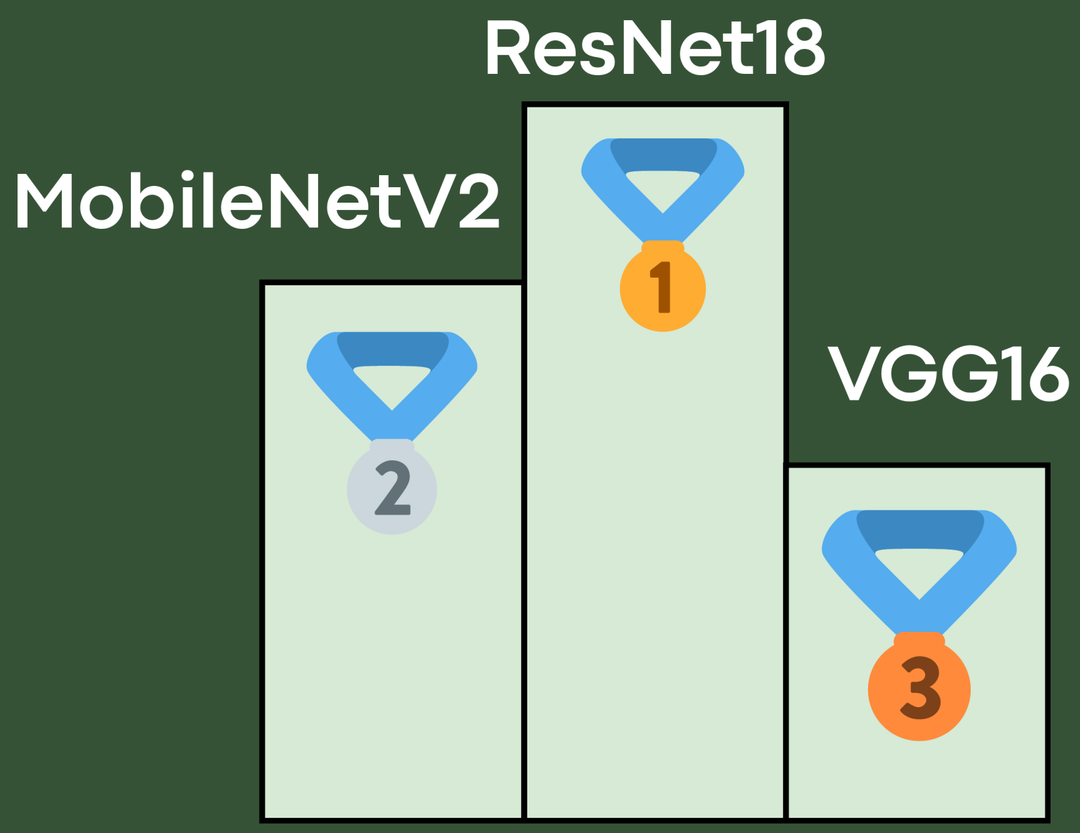
5. Model Selection
   - Used three CNN architectures:
     - VGG16 (deep but heavy)
     - ResNet18 (residual blocks for gradient stability)
     - MobileNetV2 (lightweight & mobile-friendly)
6. Transfer Learning & Fine-Tuning
   - Pretrained Models (on ImageNet)
   - Frozen convolutional layers
   - Replaced final classifier with:
     - Hidden layers (customized)
     - Output layer (50 classes)
   - MobileNetV2: unfreeze last 6 layers for more flexible learning
7. Training & Evaluation
   - Optimizer: Adam
   - Loss Function: CrossEntropyLoss
   - Epochs:
     - 10 (MobileNetV2, ResNet18)
     - 20 (VGG16)
   - Metrics Tracked:
     - Accuracy, Precision, Recall, F1-Score
     - Training vs. Validation Loss

Results



Metric	MobileNetV2	ResNet18	VGG16
Train Accuracy	98.04%	99.87%	98.93%
Validation Accuracy	91.68%	93.95%	84.50%
Test Accuracy	92.06%	95.15%	83.36%
Precision	0.9832	0.9645	0.923
Recall	0.9321	0.9621	0.9128
F-1 Score	0.9319	0.9619	0.9128
Inference Time/ batch	0.0130 sec	0.0053 sec	0.0033 sec
Max GPU memory allocated	1289.65 MB	3306.50 MB	8038.96 MB

Conclusion



1. ResNet18 is the Best Overall
   - Achieved highest test accuracy (95.15%) and F1-score (96.19%)
   - Balanced performance across all metrics
   - Efficient enough for mid-range devices(i.e. with low memory GPU)

2. MobileNetV2: Lightweight, Solid Performance
   - Great choice for edge/mobile deployment
   - Accuracy (92.06%) with minimal GPU usage (1289MB)
   - Best efficiency-to-performance ratio

3. VGG16: High Capacity
   - Good training accuracy (98.93%)
   - Prone to overfitting, lower test accuracy (83.86%)
   - Very resource-heavy (GPU: 8038MB)

THANKYOU