DOCUMENTATION

“Tree Species Classification

Dataset Used: “Leaf Images Dataset” by ichhadhari (Link)

Comparing DenseNet, ResNet, and Xception:

1. DenseNet

Key Features:

- **Dense Connections**: Each layer is connected to every other layer, ensuring maximum information flow.
- **Gradient Flow**: Improved gradient flow due to direct connections between all layers.
- **Feature Reuse**: Earlier features are reused in later layers, reducing redundancy.
- **Parameter Efficiency**: Requires fewer parameters than comparable architectures, as no redundant feature maps are learned.

Strengths:

- Handles small datasets well by leveraging feature reuse, making it suitable for datasets like **Leaf Images Dataset** with potentially limited samples.
- Excellent at capturing fine-grained details, such as leaf vein patterns and edges.

Weaknesses:

- High memory consumption due to dense connections.
- Computationally expensive compared to simpler architectures.

Suitability for Tree Classification:

DenseNet's ability to extract intricate details makes it well-suited for classifying tree species based on leaf morphology.

Model metrics before and after finetuning:

Classification Report:

Before:

Classification Report:

	precision	recall	f1-score	support
Vad	0.94	0.88	0.91	72
Indian Rubber Tree	1.00	0.97	0.99	69
Sonmohar	0.95	0.96	0.96	57
Vilayati Chinch	0.69	0.82	0.75	56
Nilgiri	0.89	0.86	0.88	59
Sita Ashok	0.98	0.97	0.98	63
Apta	0.83	0.89	0.86	56
Kashid	0.88	0.82	0.85	55
Karanj	1.00	0.88	0.93	57
Pimpal	0.87	0.95	0.91	56
accuracy			0.90	600
macro avg	0.90	0.90	0.90	600
weighted avg	0.91	0.90	0.90	600

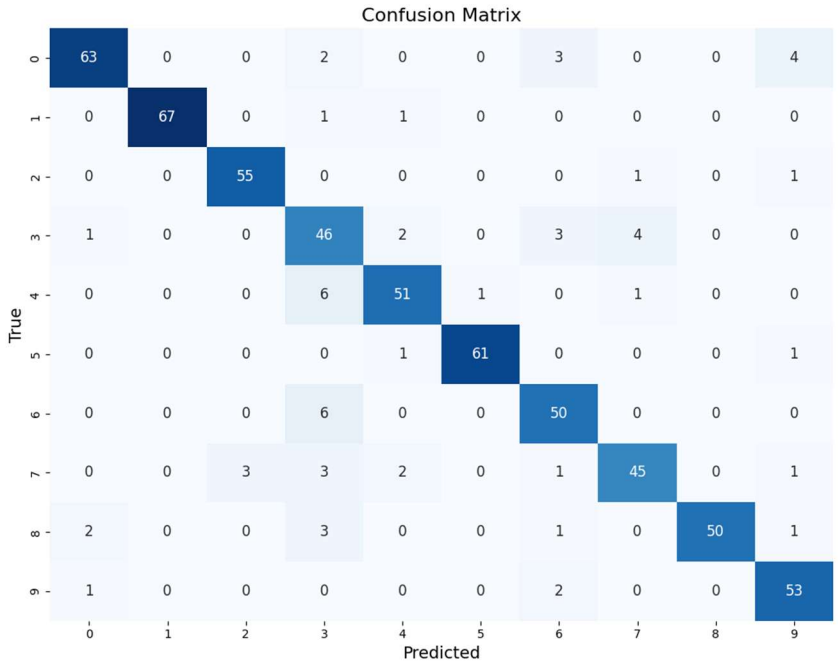
After:

Classification Report:

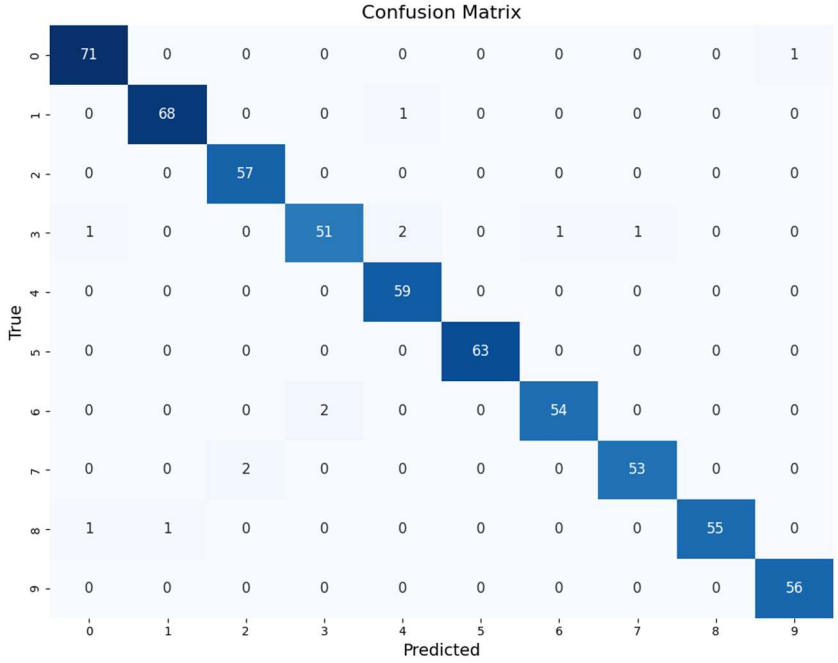
	precision	recall	f1-score	support
Vad	0.97	0.99	0.98	72
Indian Rubber Tree	0.99	0.99	0.99	69
Sonmohar	0.97	1.00	0.98	57
Vilayati Chinch	0.96	0.91	0.94	56
Nilgiri	0.95	1.00	0.98	59
Sita Ashok	1.00	1.00	1.00	63
Apta	0.98	0.96	0.97	56
Kashid	0.98	0.96	0.97	55
Karanj	1.00	0.96	0.98	57
Pimpal	0.98	1.00	0.99	56
accuracy			0.98	600
macro avg	0.98	0.98	0.98	600
weighted avg	0.98	0.98	0.98	600

Confusion Matrix:

Before

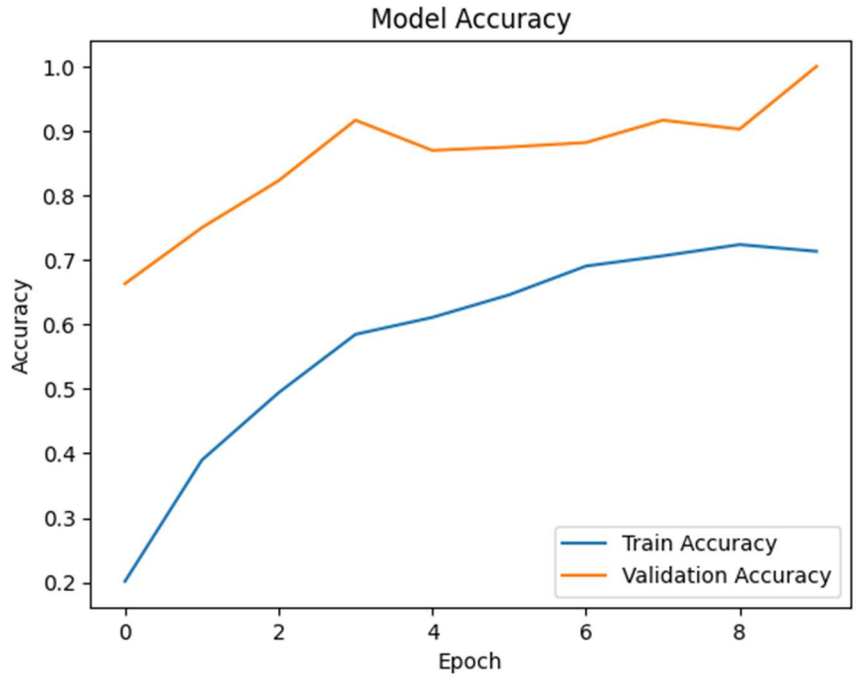


After:

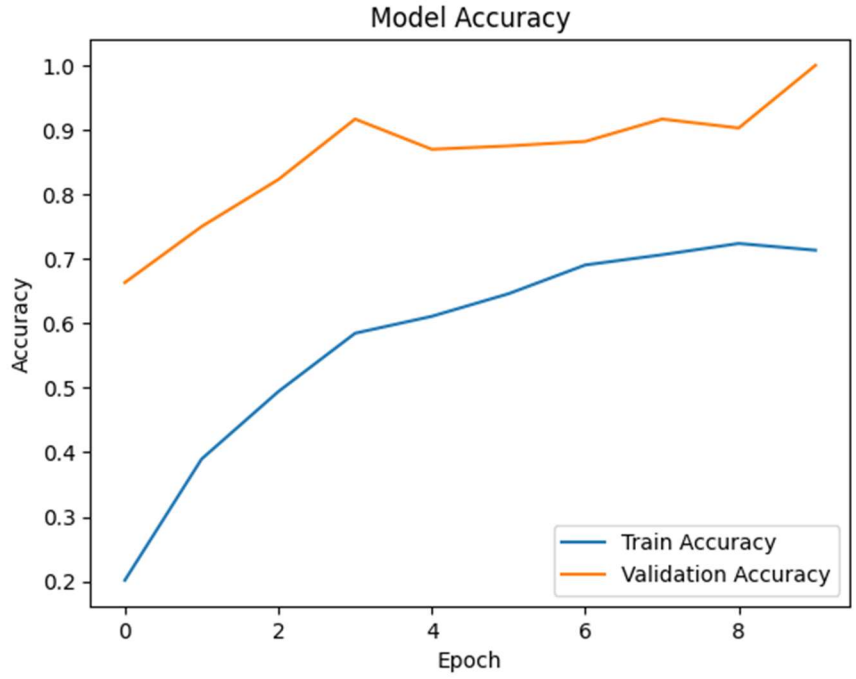


Model Accuracy:

Before:

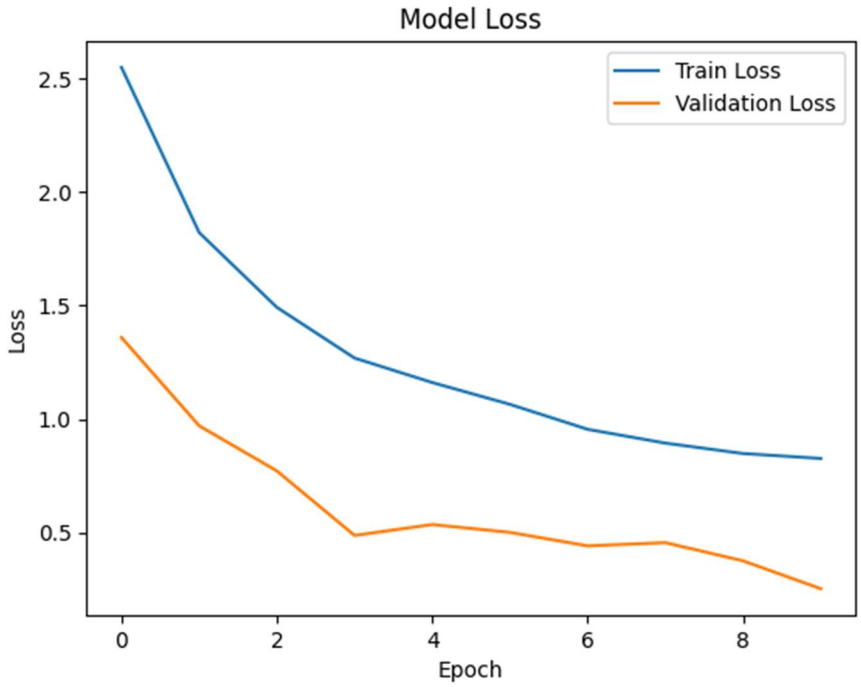


After:

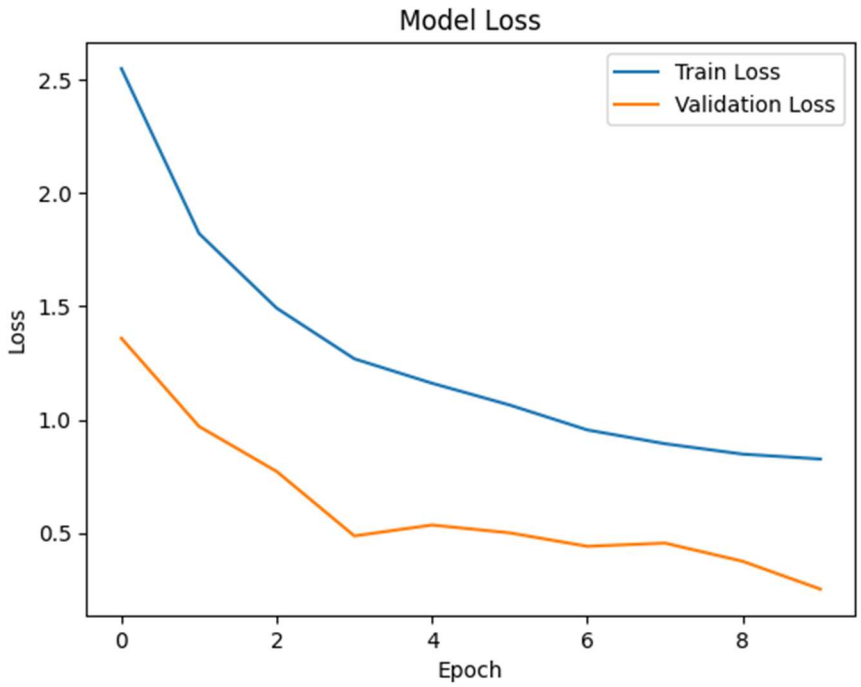


Model Loss:

Before:

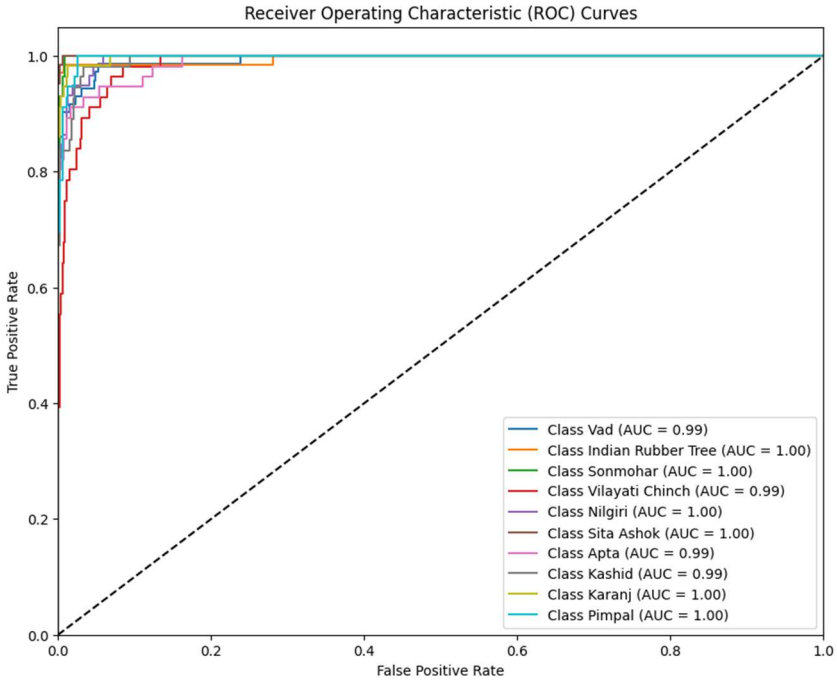


After:

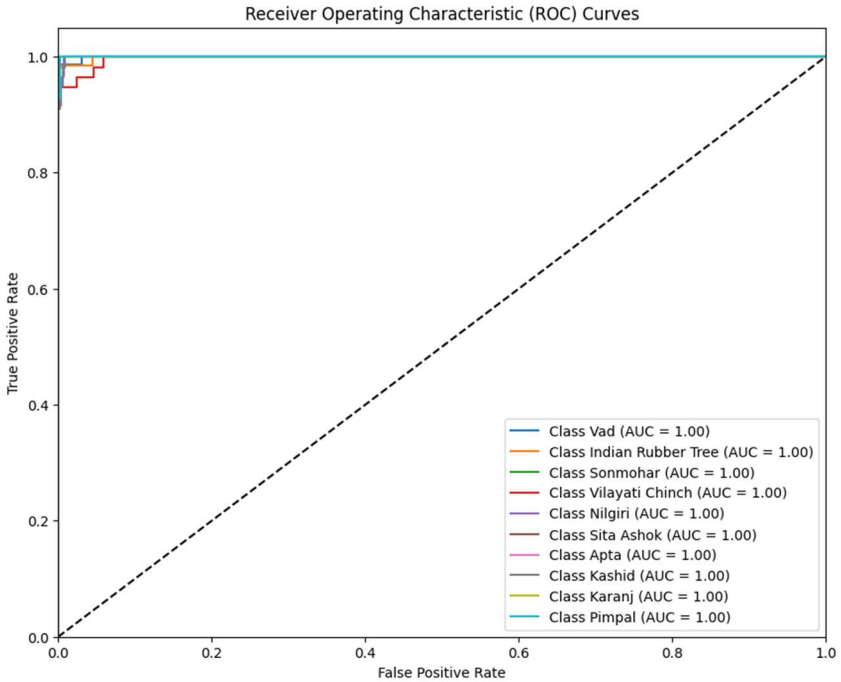


ROC curve:

Before:



After:



2. ResNet (Residual Networks)

Key Features:

- **Residual Connections**: Introduces skip connections, which allow gradients to flow more easily through deeper networks.
- **Vanishing Gradient Solution**: Can be trained very deep without vanishing gradients, leading to high performance.
- **Versatility**: Adaptable to a wide range of tasks.

Strengths:

- Works well on both large and small datasets, providing a balance of performance and computational efficiency.
- Faster convergence during training due to skip connections.

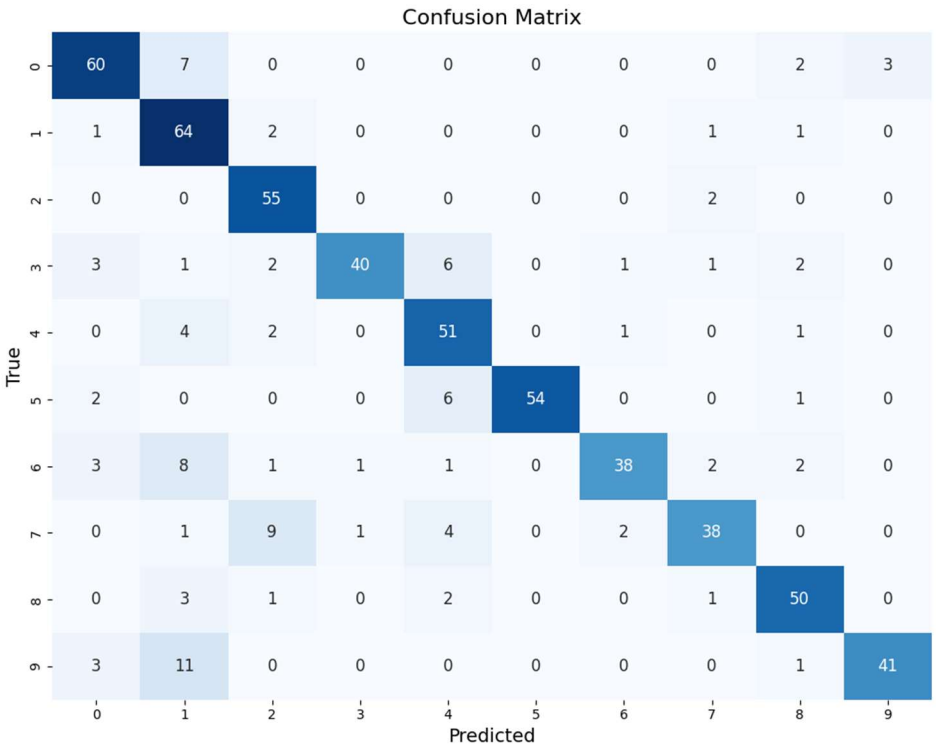
Weaknesses:

- Less parameter-efficient than DenseNet because features are not reused as extensively.
- May not capture the finest details as well as DenseNet.

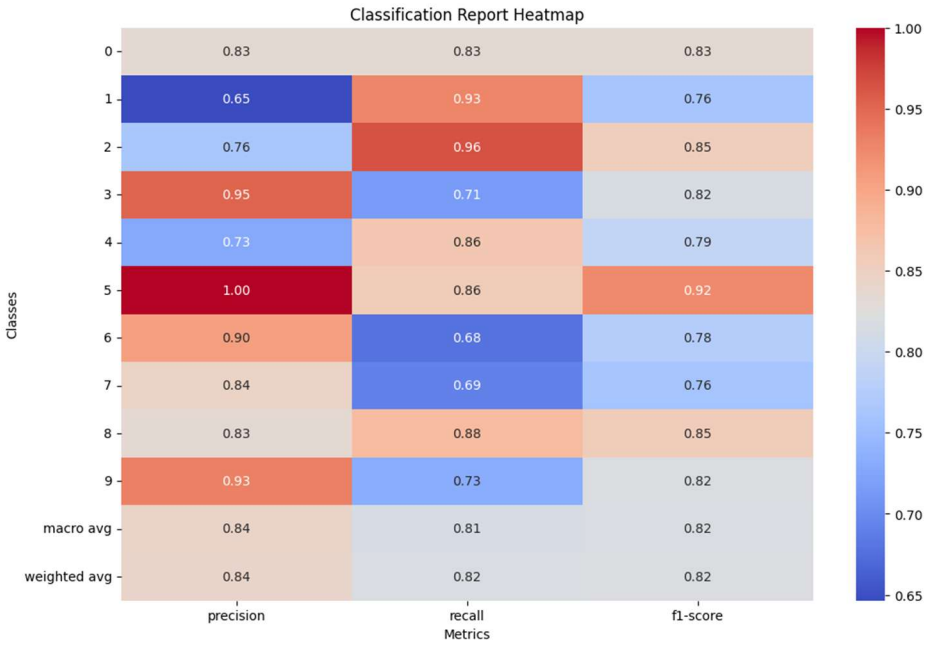
Suitability for Tree Classification:

ResNet is robust for general classification tasks and can perform well on datasets like **Flavia**, though its lack of intricate feature reuse may slightly limit performance for fine-grained tasks like leaf vein analysis

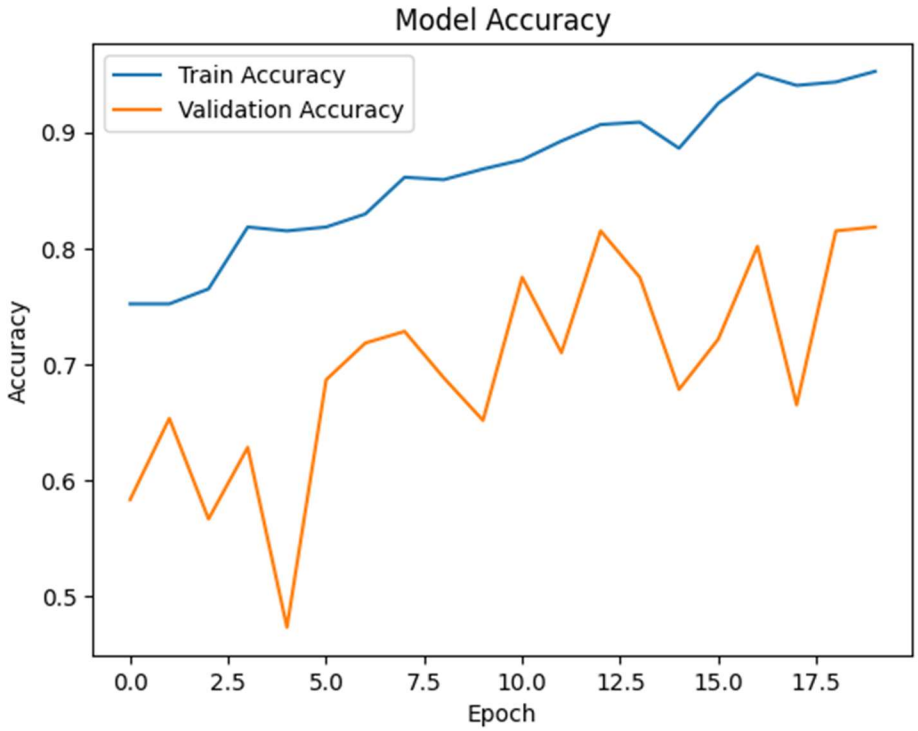
Confusion Matrix:



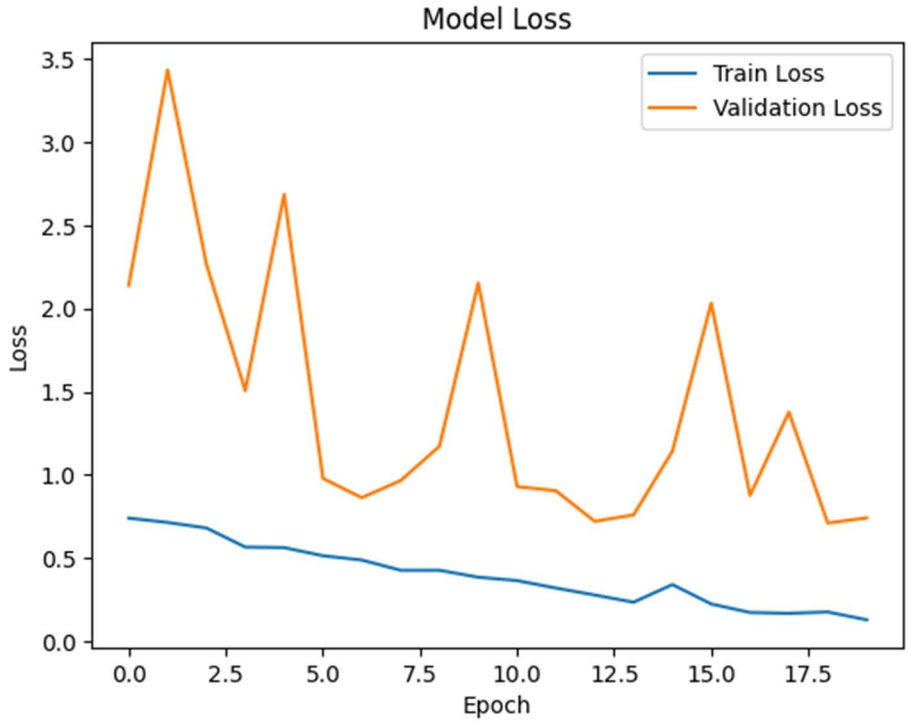
Classification Report:



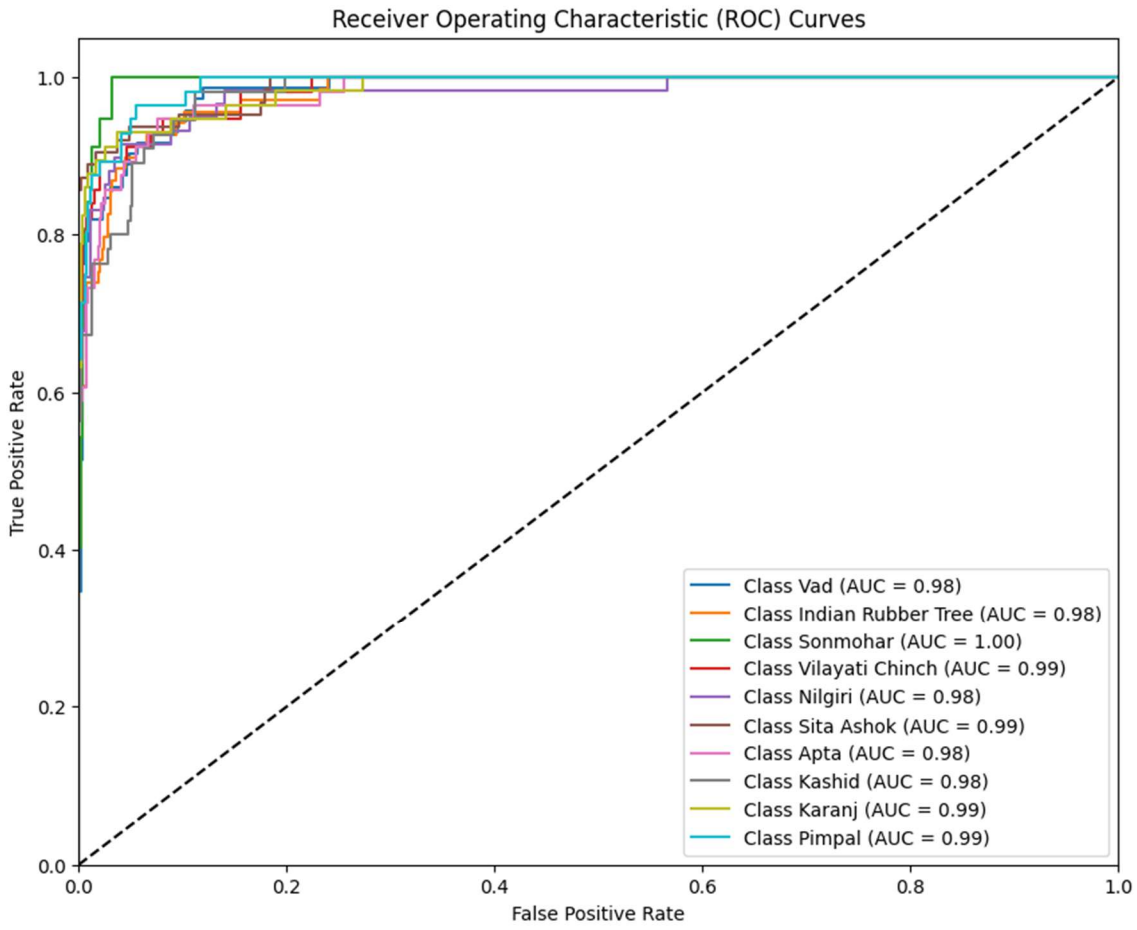
Model Accuracy:



Model Loss:



ROC curve:


Receiver Operating Characteristic (ROC) Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class Vad (AUC = 0.98)
Class Indian Rubber Tree (AUC = 0.98)
Class Sonmohar (AUC = 1.00)
Class Vilayati Chinch (AUC = 0.99)
Class Nilgiri (AUC = 0.98)
Class Sita Ashok (AUC = 0.99)
Class Apta (AUC = 0.98)
Class Kashid (AUC = 0.98)
Class Karanj (AUC = 0.99)
Class Pimpal (AUC = 0.99)

3. Xception

Key Features:

- **Depthwise Separable Convolutions**: Decomposes standard convolutions into depth wise and pointwise convolutions, drastically reducing computational cost.
- **Efficiency and Performance**: Combines the strengths of inception modules and deep separable convolutions.
- **Deep and Efficient**: Allows for deeper architectures with fewer parameters.

Strengths:

- High efficiency in terms of computational resources.
- Excels at capturing spatial relationships in images, which can be helpful for identifying complex textures like leaf surfaces.

Weaknesses:

- Requires more training data to fully leverage its depth and complexity.
- May struggle with smaller datasets if not paired with appropriate data augmentation.

Suitability for Tree Classification:

Xception's focus on spatial relationships makes it effective for leaf classification, particularly for recognizing unique textures. However, it may need more extensive pretraining or fine-tuning on datasets like **Fossil Leaf** or **Flavia**.

Model result before and after finetuning:

Classification Report:

Before:

Classification Report:

	precision	recall	f1-score	support
Vad	0.92	0.99	0.95	72
Indian Rubber Tree	0.98	0.94	0.96	69
Sonmohar	1.00	0.95	0.97	57
Vilayati Chinch	0.95	0.95	0.95	56
Nilgiri	0.95	0.92	0.93	59
Sita Ashok	0.95	0.97	0.96	63
Apta	0.98	0.91	0.94	56
Kashid	0.83	0.98	0.90	55
Karanj	0.95	0.93	0.94	57
Pimpal	0.94	0.89	0.92	56
accuracy			0.94	600
macro avg	0.95	0.94	0.94	600
weighted avg	0.95	0.94	0.94	600

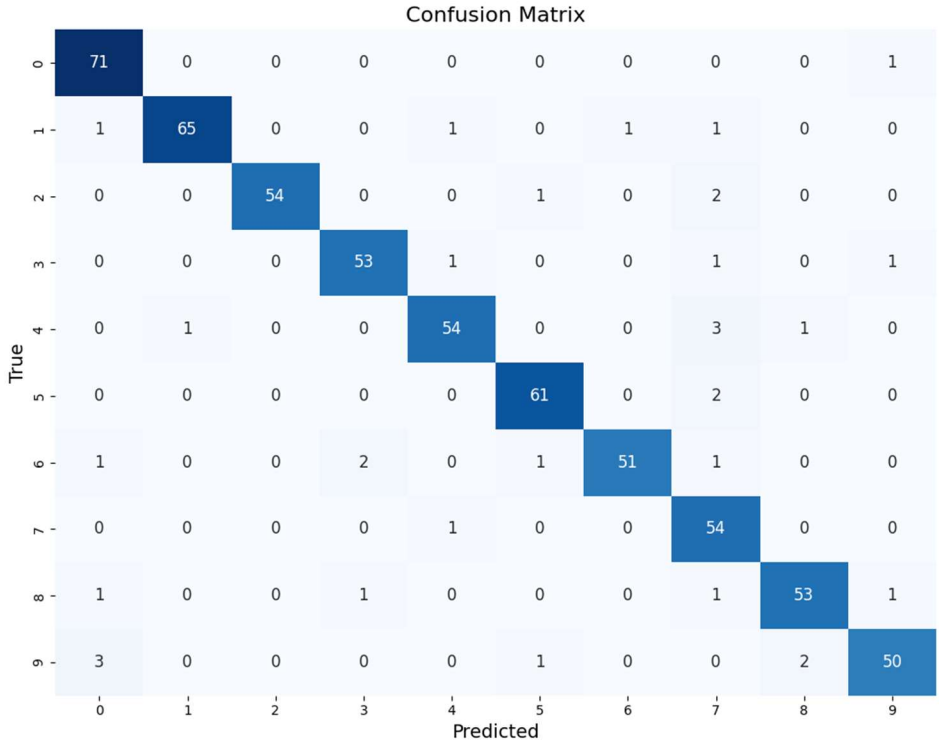
After:

Classification Report:

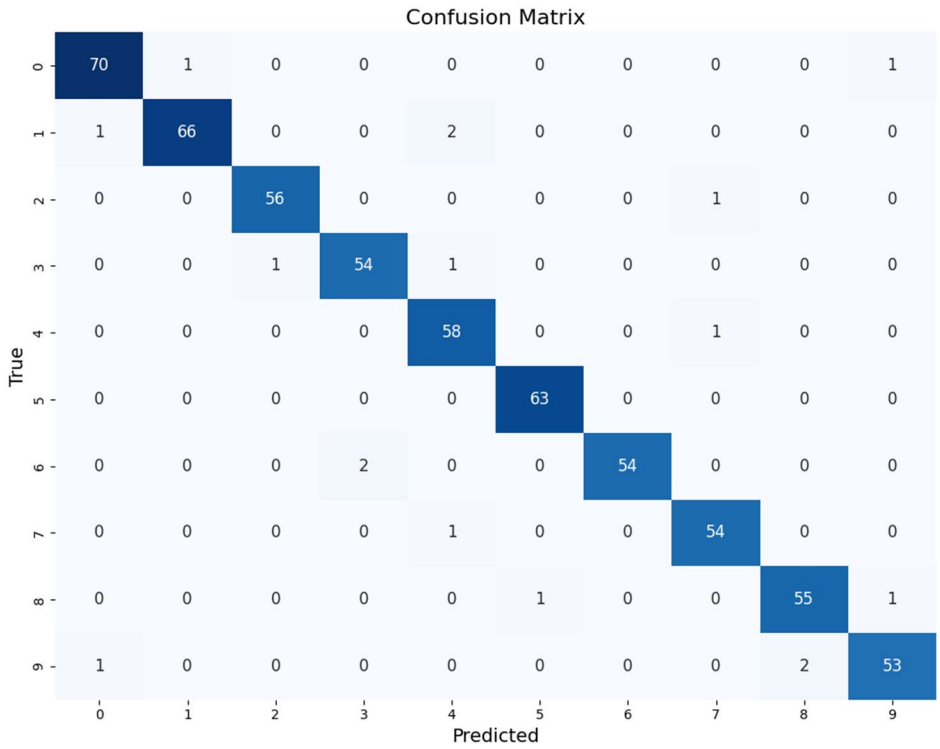
	precision	recall	f1-score	support
Vad	0.97	0.97	0.97	72
Indian Rubber Tree	0.99	0.96	0.97	69
Sonmohar	0.98	0.98	0.98	57
Vilayati Chinch	0.96	0.96	0.96	56
Nilgiri	0.94	0.98	0.96	59
Sita Ashok	0.98	1.00	0.99	63
Apta	1.00	0.96	0.98	56
Kashid	0.96	0.98	0.97	55
Karanj	0.96	0.96	0.96	57
Pimpal	0.96	0.95	0.95	56
accuracy			0.97	600
macro avg	0.97	0.97	0.97	600
weighted avg	0.97	0.97	0.97	600

Confusion Matrix:

Before:

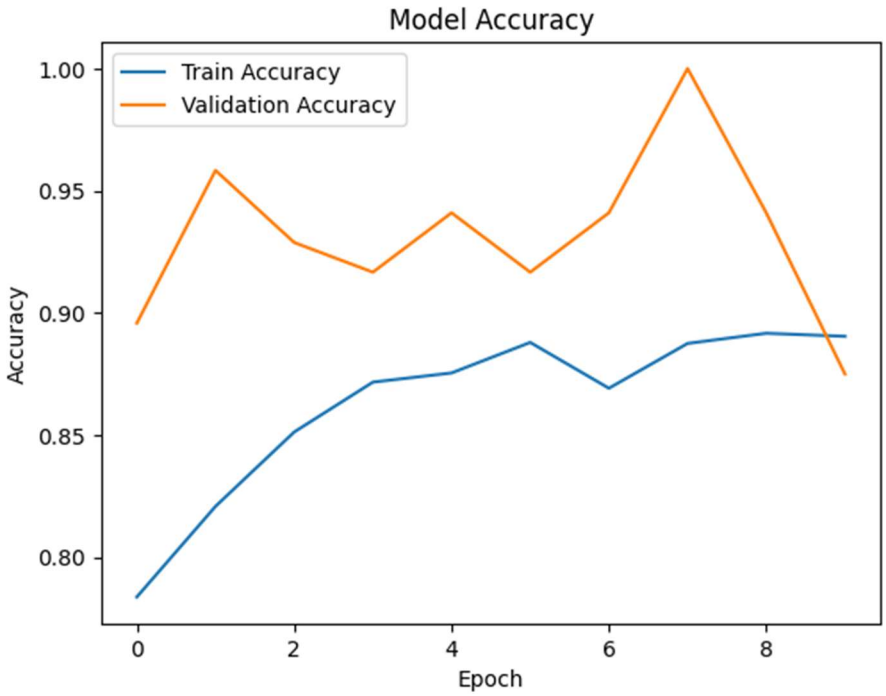


After:

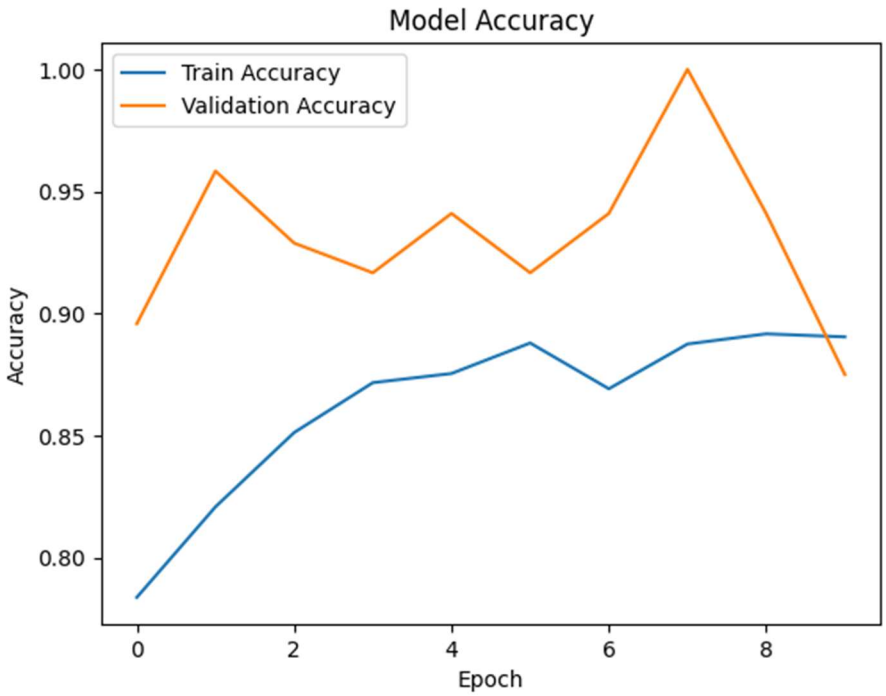


Model Accuracy:

Before:

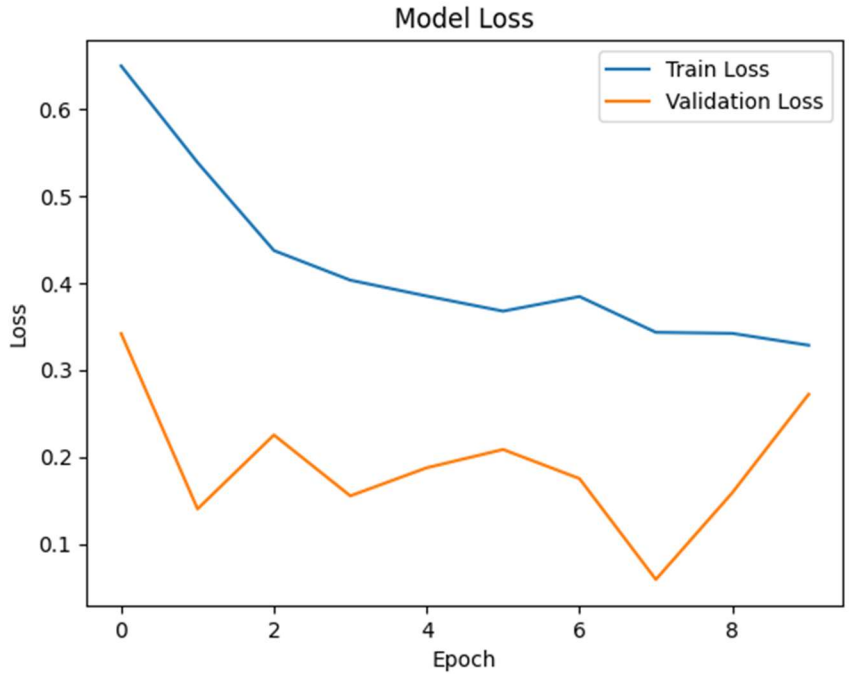


After:

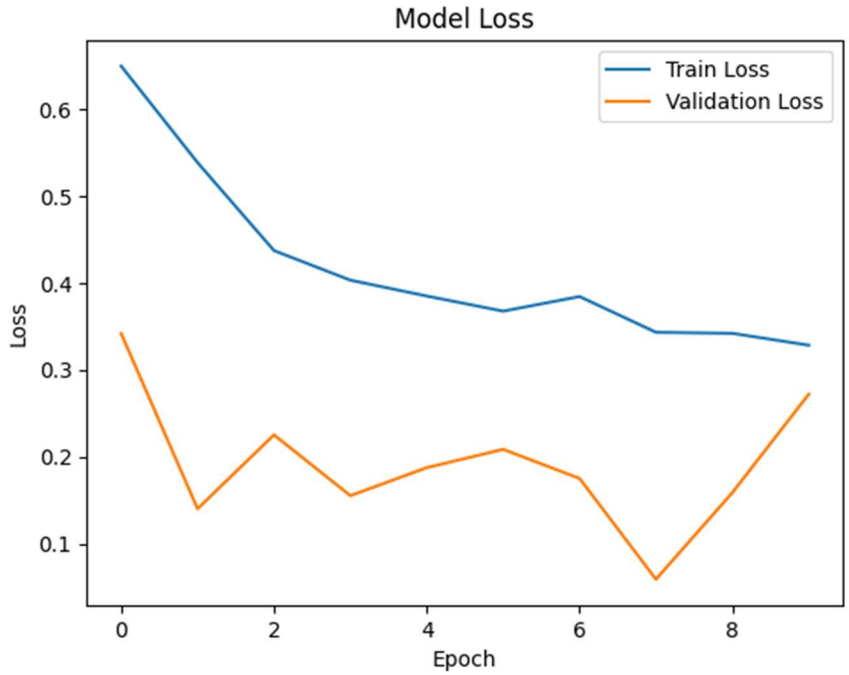


Model Loss:

Before:

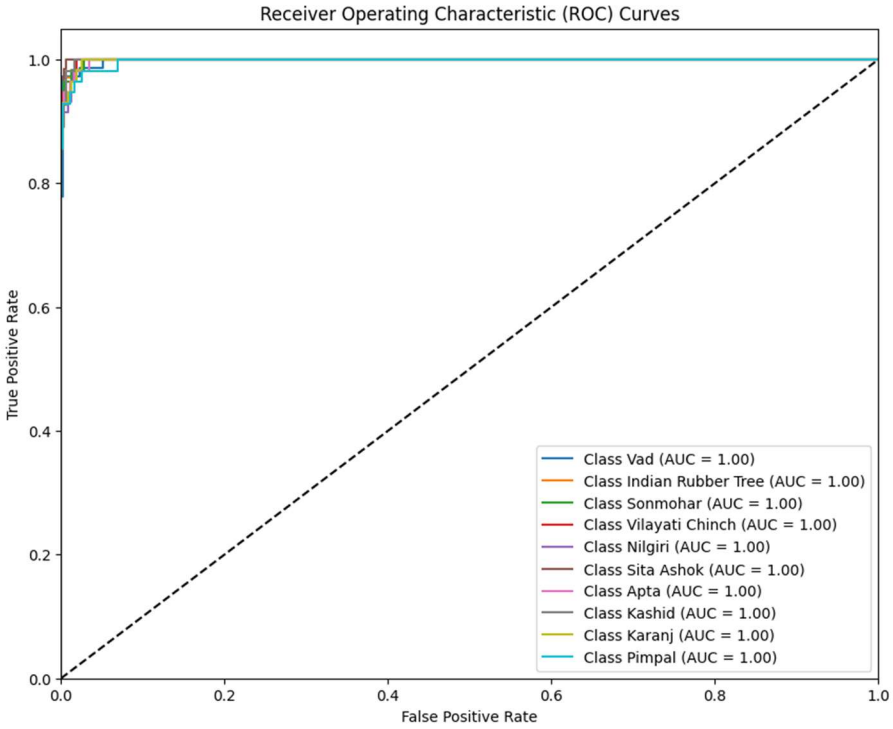


After:

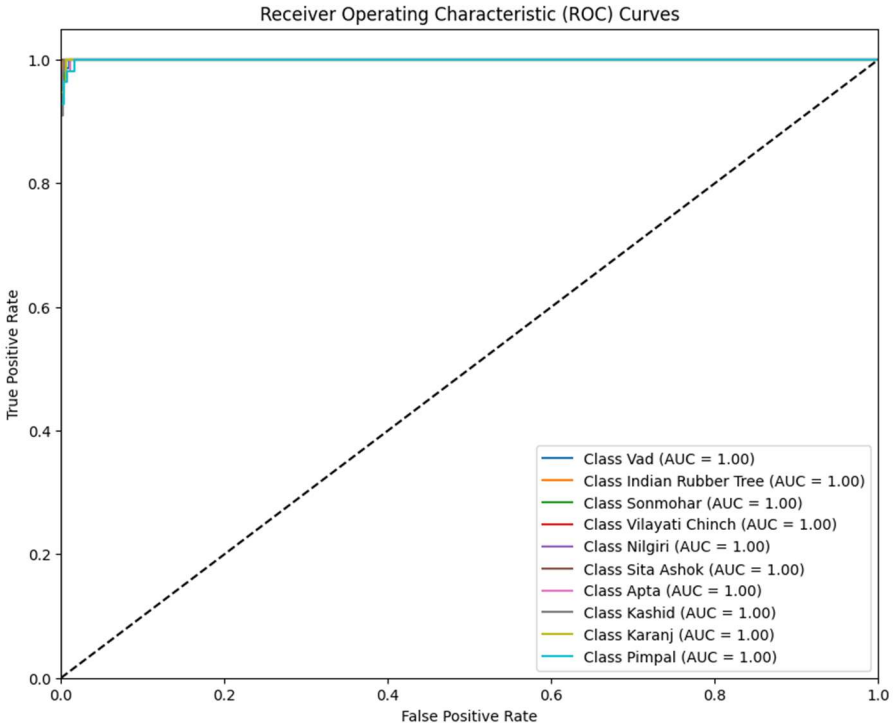


ROC curve:

Before:



After:



Papers for three models :

1. DenseNet121:

[1608.06993] Densely Connected Convolutional Networks

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			

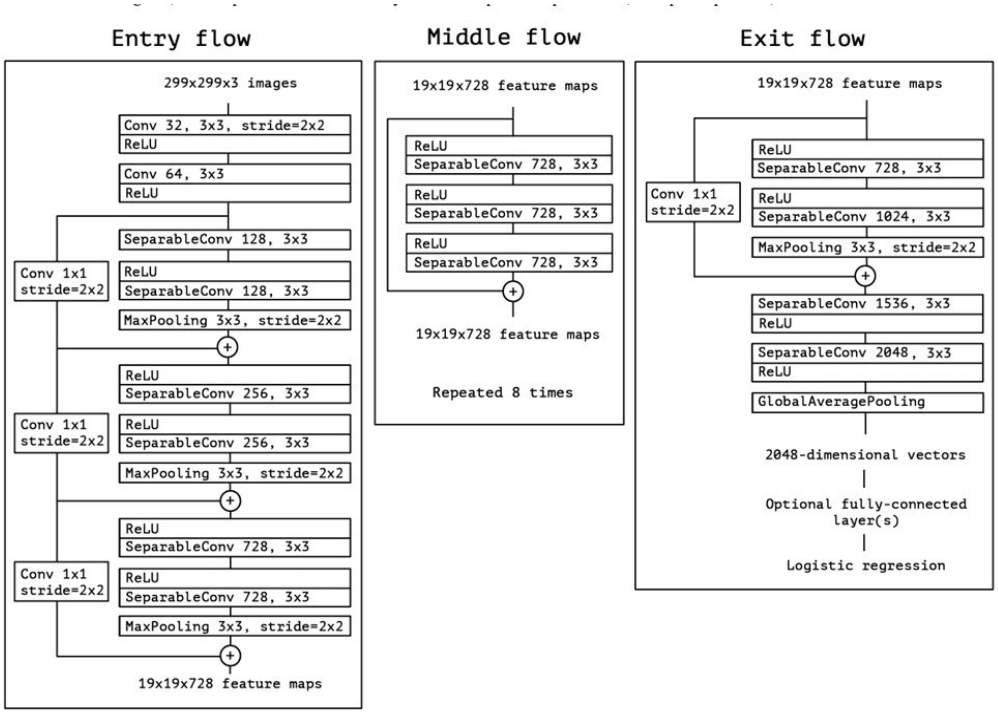
2. ResNet50:

https://arxiv.org/pdf/1512.03385

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				
		[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

3. Xception:

[1610.02357] Xception: Deep Learning with Depthwise Separable Convolutions



Comparison of the three models is conducted based on their results

Densenet121	**98%**
ResNet50	**82%**
Xception	**97%**

Architecture	Pros	Cons
DenseNet-121	Highly efficient in parameters, improved gradient flow, compact representations	High computation and memory cost, potential overfitting on small datasets
ResNet-50	Stable training, widely supported, versatile, and effective for transfer learning	Computationally and memory intensive, diminishing returns with depth
Xception	Parameter-efficient, excellent performance, strong generalization	Computationally expensive, less adoption for transfer learning, more complex to implement

Explaining the advantages of a specific architecture for a given task and dataset requires aligning the architecture's strengths with the task's requirements and the dataset's characteristics

Key Considerations

1. **Task Requirements**
    a. **Image Classification:** High accuracy, feature extraction, and generalization ability.
    b. **Object Detection/Segmentation:** Contextual and spatial awareness with efficient computation.
    c. **Resource Constraints:** Limited compute or memory may prioritize lighter architectures.
2. **Dataset Characteristics**
    a. **Size:** Smaller datasets may favor architectures with fewer parameters to prevent overfitting.
    b. **Complexity:** Complex datasets with diverse features may benefit from deeper or more advanced architectures.
    c. **Resolution:** High-resolution datasets may favor architectures optimized for finer details.

1. DenseNet-121 Advantages

Best Suited For: Small to medium-sized datasets or tasks requiring efficient parameter usage and gradient flow.

Advantages:

- **Compact Representations:** Dense connections reuse features, making it highly efficient for datasets with limited samples.
- **Reduced Overfitting:** Smaller parameter count minimizes overfitting risks for smaller datasets.
- **Gradient Flow:** Efficient gradient propagation ensures stable training, even with deeper networks.

2. ResNet-50 Advantages

Best Suited For: Medium to large datasets, tasks requiring strong generalization, and resource-limited environments.

Advantages:

- **Robust Generalization:** Residual connections ensure stable learning even for very deep networks, making it effective for complex datasets.
- **Transfer Learning:** Extensive pre-trained models on ImageNet make it ideal for finetuning on moderately sized datasets.
- **Efficiency:** While deep, ResNet-50 strikes a balance between computational cost and accuracy, making it practical for most tasks.
- **Versatility:** Performs well across diverse tasks like classification, detection, and segmentation.

3. Xception Advantages

Best Suited For: High-complexity datasets with large-scale or fine-grained image classification tasks.

Advantages:

- **Parameter Efficiency:** Depthwise separable convolutions reduce the model size while maintaining high performance.
- **High Accuracy:** Outperforms other architectures of similar complexity on large datasets with rich features.
- **Feature Extraction:** Excels at capturing detailed and nuanced features, making it ideal for high-resolution datasets.

Choosing the Right Architecture

To decide on the best architecture:

- **DenseNet-121:** For tasks with smaller datasets or when memory efficiency is a priority.
- **ResNet-50:** If generalization and versatility are key, especially for datasets of medium-to-large size.
- **Xception:** When the dataset is large and complex, and fine-grained details are crucial.