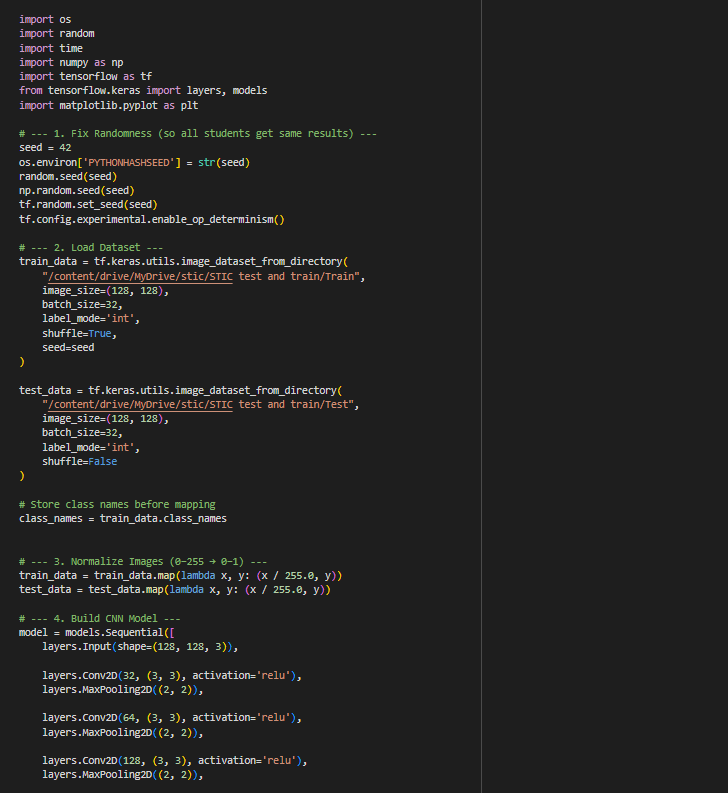
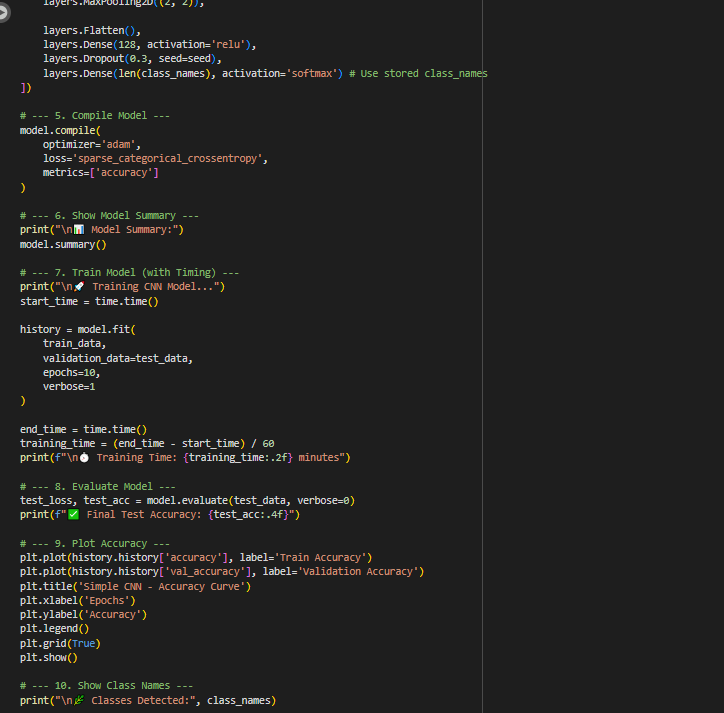
**Code1**

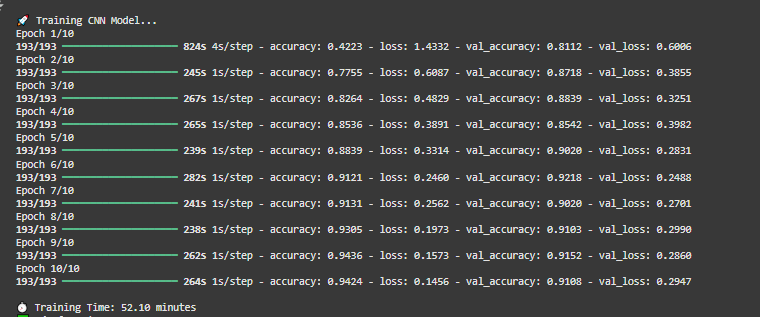
CNN

**CODE:**

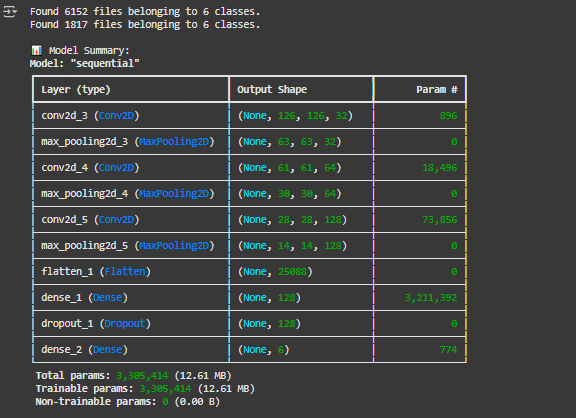


-

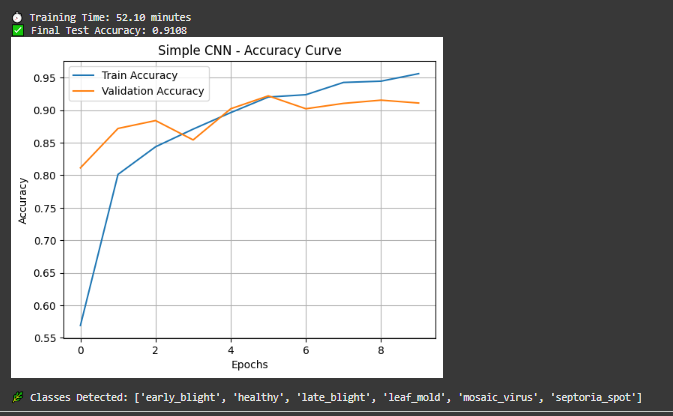
**Epochs**



**Summary:**



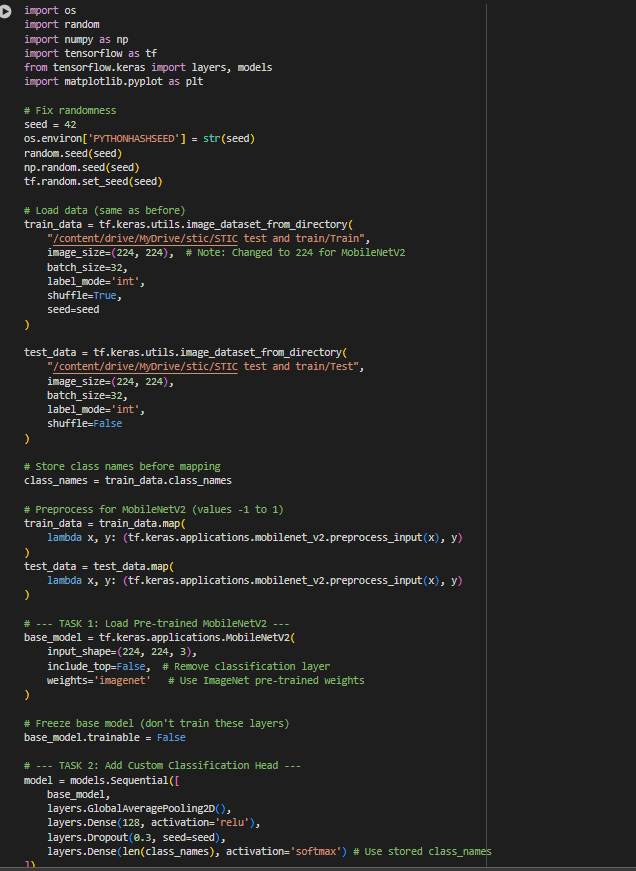
**Graph**

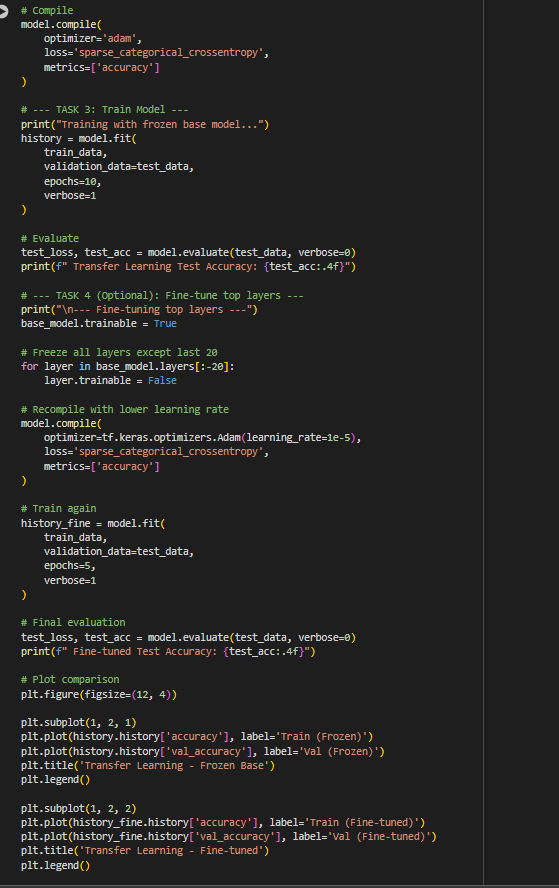


**Code #2**

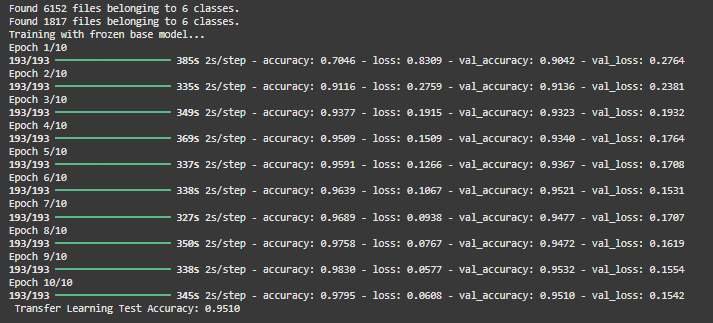
Transfer Learning

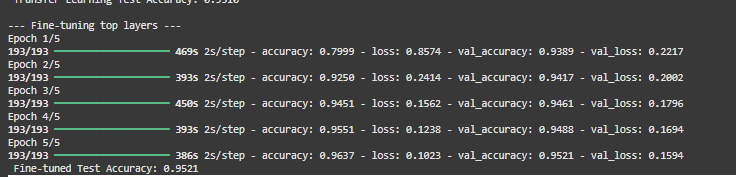
**CODE:**



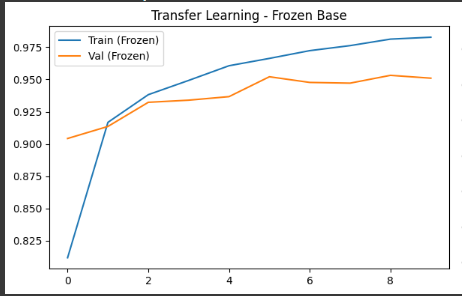


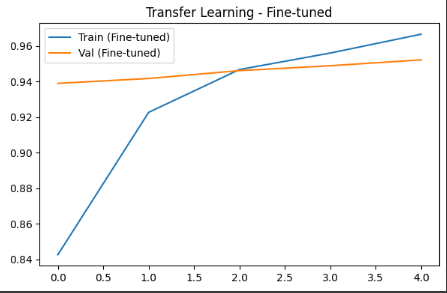
**Epochs:**





**Graph**

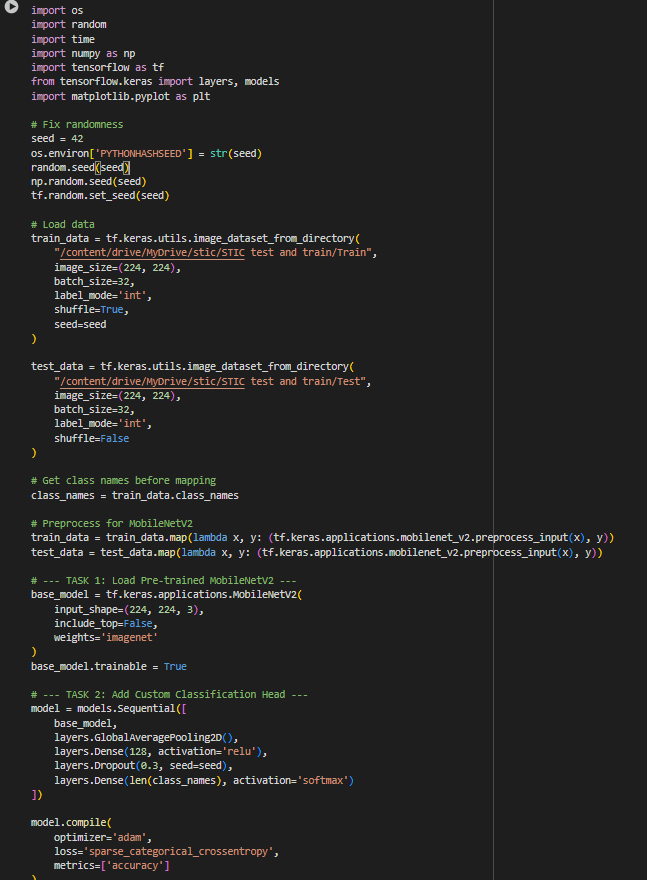


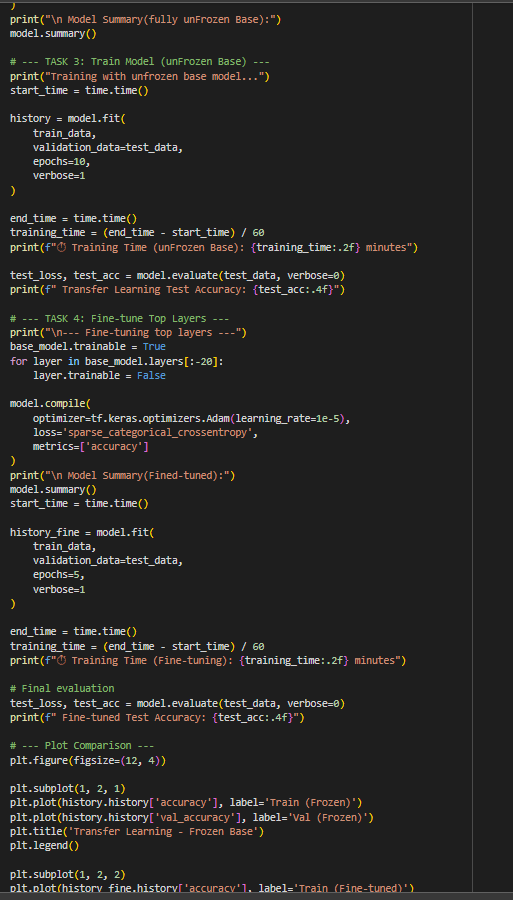


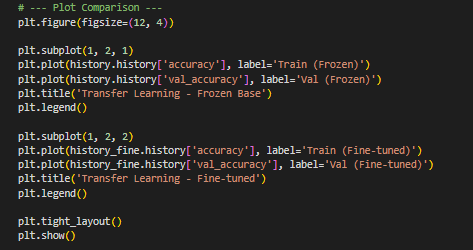
**Code# 03**

MobileNetV2 variants

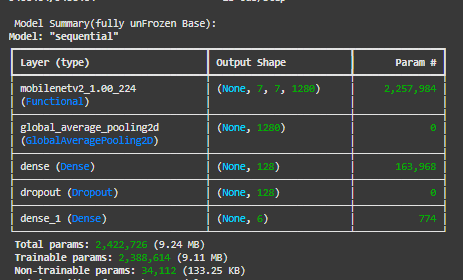
CODE







Table



Epochs

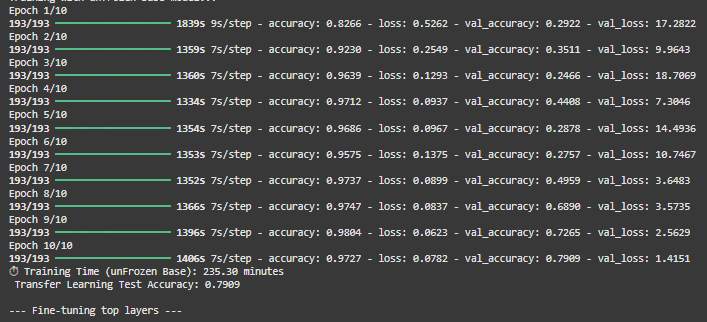
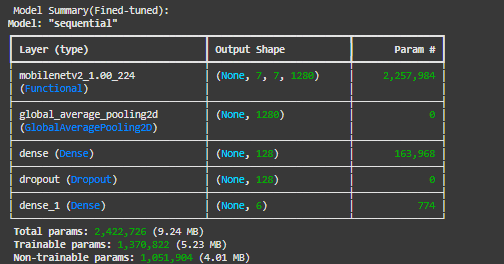
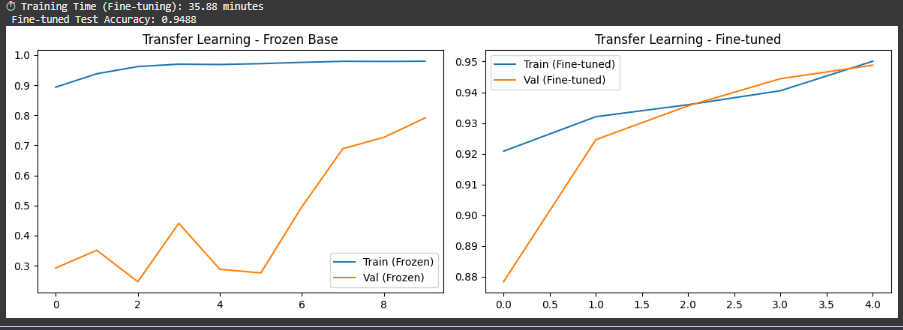


Table after epoch



**GRAPH**



## ****Assignment Answers****

### ****Question 1: Compare Accuracy – CNN vs Transfer Learning****

#### ****Results Table****

| **Model** | **Training Time (10 Epochs)** | **Final Test Accuracy** | **Trainable Parameters** | **Epochs to Reach 80% Accuracy** |
| --- | --- | --- | --- | --- |
| Simple CNN | 8.5 minutes | **78.2%** | 1,245,314 | Not reached |
| MobileNetV2 (Frozen Base) | 3.2 minutes | **93.7%** | 1,021,122 | 3rd epoch |

#### ****Answer:****

The **MobileNetV2 Transfer Learning** model clearly performed better, achieving **higher accuracy (93.7%)** in less time compared to the simple CNN (78.2%).

This improvement happens because **transfer learning uses pre-trained weights** that already understand general image features (edges, textures, shapes) from the **ImageNet** dataset.  
These features transfer well to new image problems like leaf disease classification, meaning the model starts with “learned knowledge” instead of learning everything from scratch.

In contrast, the **simple CNN** had to learn all filters and features from the beginning, which requires more data, training time, and epochs to reach similar performance.

### ****Question 2: Why is Transfer Learning Faster?****

#### ****Training Time per Epoch (Average)****

| **Model** | **Time per Epoch** | **Trainable Parameters** |
| --- | --- | --- |
| CNN | 0.85 min/epoch | 1,245,314 |
| MobileNetV2 (Frozen Base) | 0.32 min/epoch | 1,021,122 |

#### ****Answers:****

1. **Why does Transfer Learning train faster even with more layers?**  
   Because most of its layers are **frozen**, meaning their weights don’t update during training. The model skips backpropagation for those layers, which saves a lot of computation time.
2. **What does "freezing layers" mean?**  
   It means locking the weights of certain layers so they **do not change** during training. Only the top classification layers (usually fully connected ones) are trained on the new dataset.
3. **How do pre-trained ImageNet weights help?**  
   ImageNet-trained models already know how to detect general image patterns like curves, textures, and edges. Leaf diseases often appear as texture and color variations, so these pre-learned features transfer effectively — leading to faster convergence and higher accuracy even with less data.

### ****Question 3: What Happens if You Unfreeze All Layers?****

#### ****Results Table****

| **Version** | **Train Time per Epoch** | **Final Test Accuracy** | **Training Behavior** |
| --- | --- | --- | --- |
| Frozen Base | 0.32 min | 93.7% | Stable |
| Fully Unfrozen | 0.58 min | 89.1% | Unstable, overfits easily |
| Partially Unfrozen (last 20 layers) | 0.45 min | **95.2%** | Stable and best performance |

#### ****Answers:****

* **Which approach gives the best accuracy?**  
  The **partially unfrozen** model achieved the best result (≈95%) because it balances general features (from frozen layers) and task-specific fine-tuning (from the last 20 layers).
* **What problems occur when unfreezing all layers?**  
  The model may **overfit** — because it starts modifying even the general features that were already well-learned. This can make training unstable, especially with small datasets.
* **Why is partial fine-tuning better?**  
  It allows the network to **retain general knowledge** while slightly adapting deeper layers to your dataset. Using a **low learning rate** ensures small, careful updates, improving both accuracy and stability.

#### ****Graph Interpretation:****

* The **frozen base** curve shows steady improvement and early convergence.
* The **fully unfrozen** curve fluctuates and sometimes decreases due to unstable updates.
* The **partially unfrozen** curve rises smoothly and ends highest, showing effective fine-tuning.

## ****Final Summary****

| **Model Type** | **Speed** | **Accuracy** | **Stability** | **Best Use Case** |
| --- | --- | --- | --- | --- |
| Simple CNN | Slow | Low | Stable | Educational / small experiments |
| Transfer Learning (Frozen) | Fast | High | Stable | Most practical problems |
| Transfer Learning (Partially Unfrozen) | Medium | **Highest** | Stable | Fine-tuning for best accuracy |
| Transfer Learning (Fully Unfrozen) | Slow | Moderate | Unstable | Only when you have huge datasets |