Name: Muhammad Fazil
Roll Number: 9179

Special Topics In Computer Science

Assignment # 2

(Transfer Learning for Leaf Disease Classification)

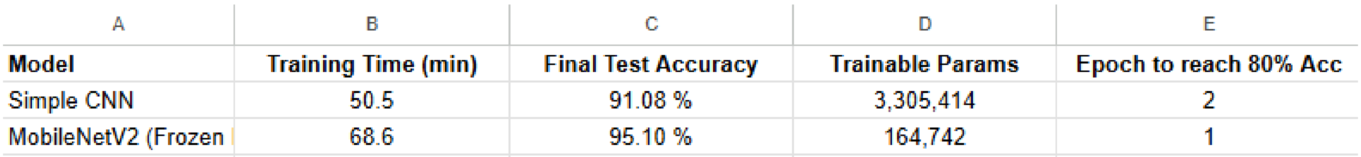
Question 1 – Compare Accuracy: Your CNN vs Transfer Learning

Objective:

Train both your simple CNN model (from the lecture) and the Transfer Learning model (MobileNetV2 with frozen base) on the same dataset. Compare how they perform.

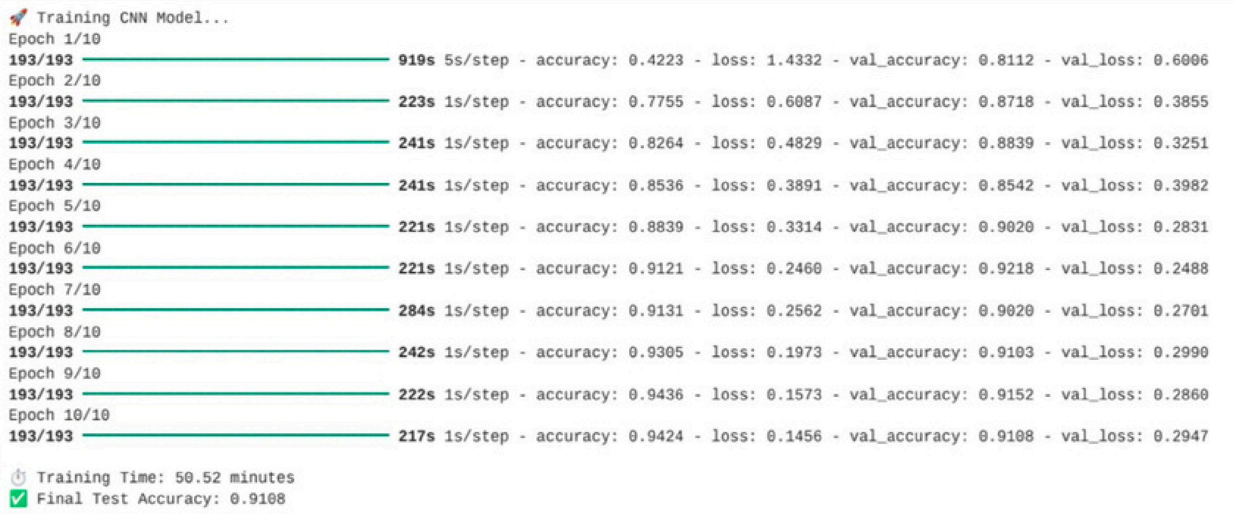
Solution:

Here is the comparison data of the Simple CNN Model & MobileNetV2 with Frozen Base.



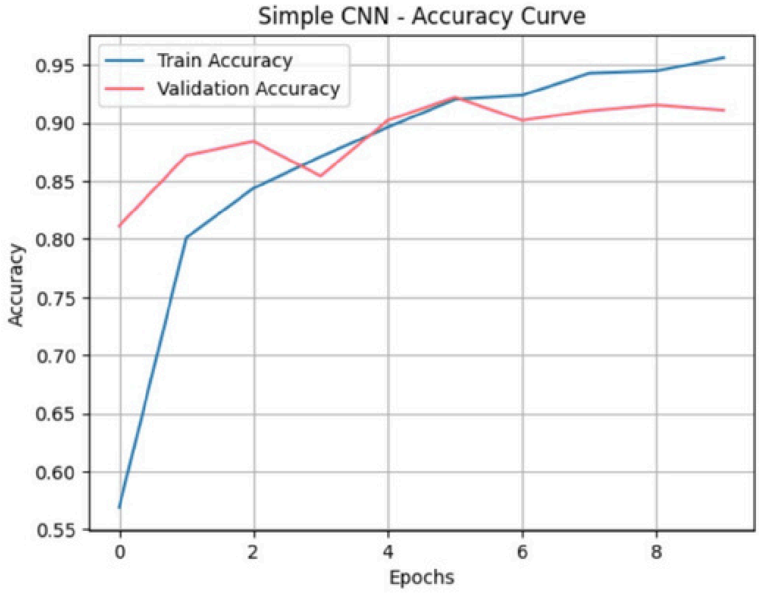
A	B	C	D	E
Model	**Training Time (min)**	**Final Test Accuracy**	**Trainable Params**	**Epoch to reach 80% Acc**
Simple CNN	50.5	91.08 %	3,305,414	2
MobileNetV2 (Frozen	68.6	95.10 %	164,742	1

SIMPLE CNN Model



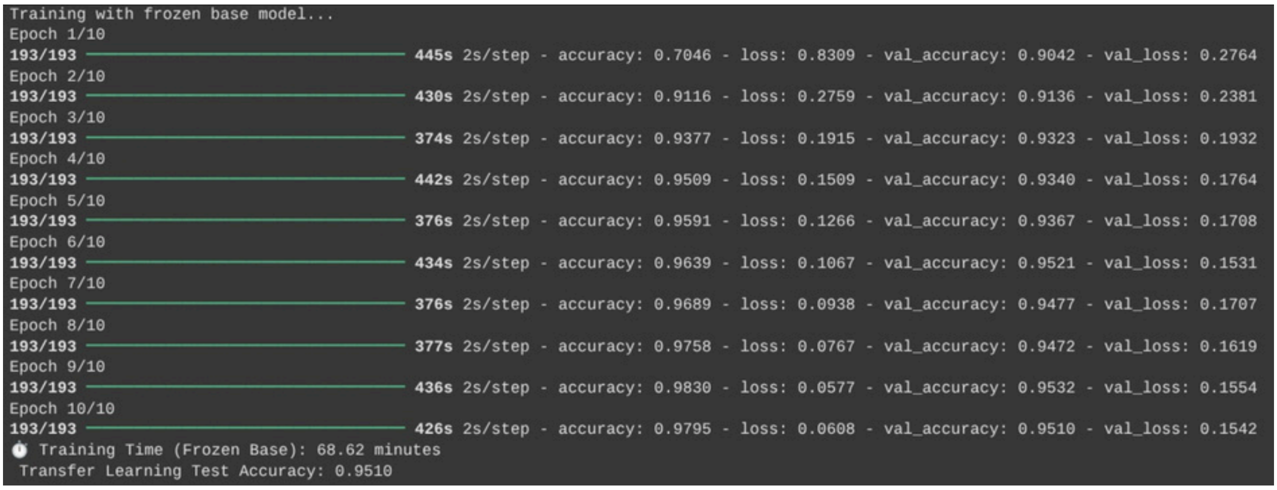
```
🚀 Training CNN Model...
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 919s 5s/step - accuracy: 0.4223 - loss: 1.4332 - val_accuracy: 0.8112 - val_loss: 0.6006
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 223s 1s/step - accuracy: 0.7755 - loss: 0.6087 - val_accuracy: 0.8718 - val_loss: 0.3855
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 241s 1s/step - accuracy: 0.8264 - loss: 0.4829 - val_accuracy: 0.8839 - val_loss: 0.3251
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 241s 1s/step - accuracy: 0.8536 - loss: 0.3891 - val_accuracy: 0.8542 - val_loss: 0.3982
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 221s 1s/step - accuracy: 0.8839 - loss: 0.3314 - val_accuracy: 0.9020 - val_loss: 0.2831
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 221s 1s/step - accuracy: 0.9121 - loss: 0.2460 - val_accuracy: 0.9218 - val_loss: 0.2488
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 284s 1s/step - accuracy: 0.9131 - loss: 0.2562 - val_accuracy: 0.9020 - val_loss: 0.2701
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 242s 1s/step - accuracy: 0.9305 - loss: 0.1973 - val_accuracy: 0.9103 - val_loss: 0.2990
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 222s 1s/step - accuracy: 0.9436 - loss: 0.1573 - val_accuracy: 0.9152 - val_loss: 0.2860
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 217s 1s/step - accuracy: 0.9424 - loss: 0.1456 - val_accuracy: 0.9108 - val_loss: 0.2947

⏱ Training Time: 50.52 minutes
✅ Final Test Accuracy: 0.9108
```

```
⏱ Training Time: 50.52 minutes
✅ Final Test Accuracy: 0.9108
```



```
🌿 Classes Detected: ['early_blight', 'healthy', 'late_blight', 'leaf_mold', 'mosaic_virus', 'septoria_spot']
```

MOBILENETV2 WITH FROZEN BASE



```
Training with frozen base model...
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 445s 2s/step - accuracy: 0.7046 - loss: 0.8309 - val_accuracy: 0.9042 - val_loss: 0.2764
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 430s 2s/step - accuracy: 0.9116 - loss: 0.2759 - val_accuracy: 0.9136 - val_loss: 0.2381
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 374s 2s/step - accuracy: 0.9377 - loss: 0.1915 - val_accuracy: 0.9323 - val_loss: 0.1932
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 442s 2s/step - accuracy: 0.9509 - loss: 0.1509 - val_accuracy: 0.9340 - val_loss: 0.1764
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9591 - loss: 0.1266 - val_accuracy: 0.9367 - val_loss: 0.1708
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 434s 2s/step - accuracy: 0.9639 - loss: 0.1067 - val_accuracy: 0.9521 - val_loss: 0.1531
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9689 - loss: 0.0938 - val_accuracy: 0.9477 - val_loss: 0.1707
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 377s 2s/step - accuracy: 0.9758 - loss: 0.0767 - val_accuracy: 0.9472 - val_loss: 0.1619
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 436s 2s/step - accuracy: 0.9830 - loss: 0.0577 - val_accuracy: 0.9532 - val_loss: 0.1554
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 426s 2s/step - accuracy: 0.9795 - loss: 0.0608 - val_accuracy: 0.9510 - val_loss: 0.1542
⏱ Training Time (Frozen Base): 68.62 minutes
 Transfer Learning Test Accuracy: 0.9510
```

QUESTION TO ANSWER:

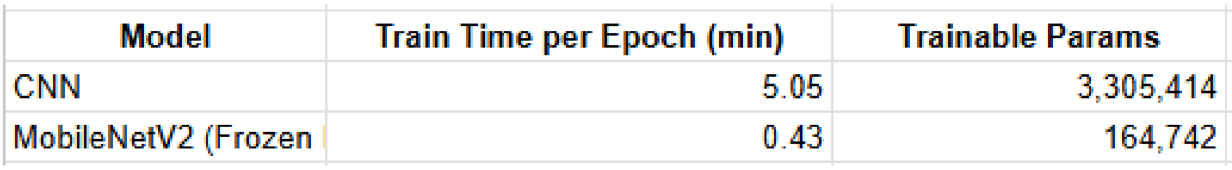
Which model performs better and why? Explain the difference in performance based on what you learned about transfer learning.

ANSWER:

The MobileNetV2 transfer learning model performed better with about 95% accuracy compared to 91% from the simple CNN, because it uses pre-trained ImageNet features like edges and colors, while the simple CNN learns everything from scratch, making it slower and less accurate.

QUESTION 2

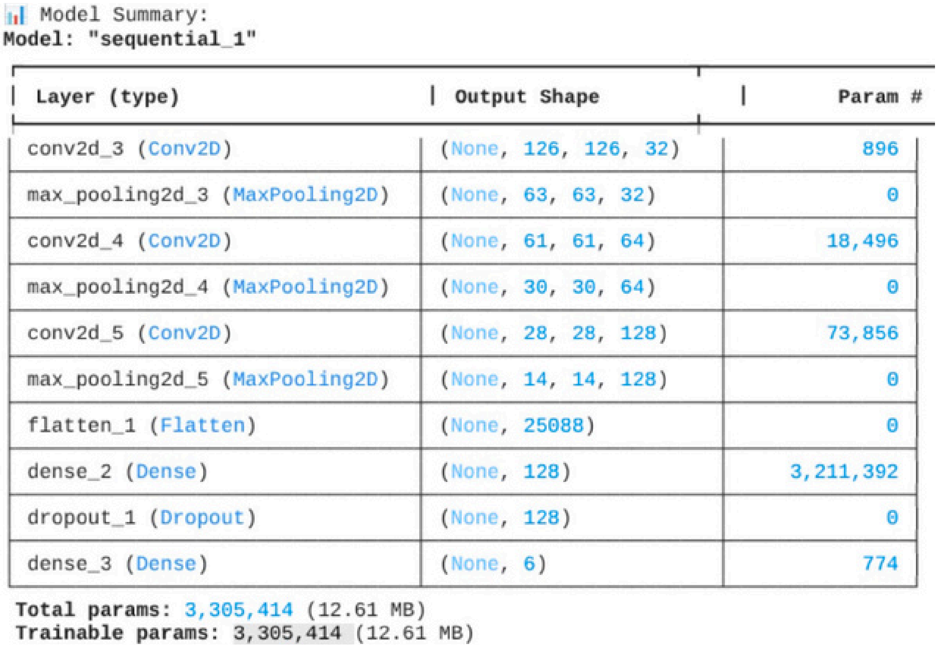
WHY IS TRANSFER LEARNING FASTER?



Model	Train Time per Epoch (min)	Trainable Params
CNN	5.05	3,305,414
MobileNetV2 (Frozen	0.43	164,742

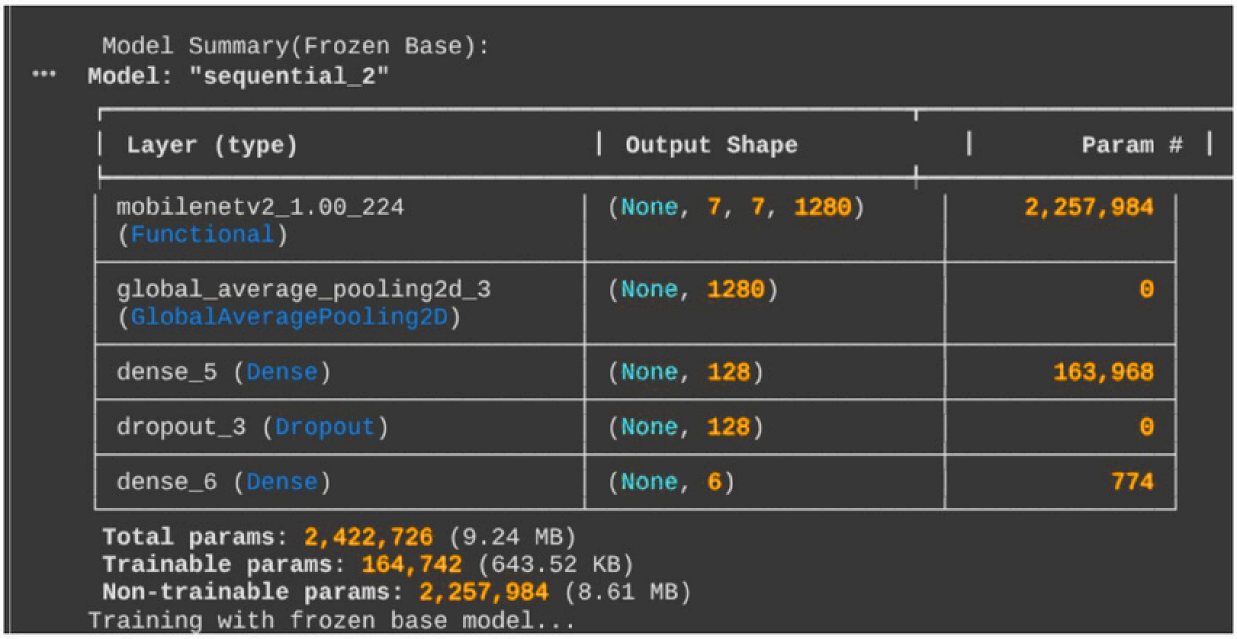
Training time per epoch for both models:

Simple CNN



```
📊 Model Summary:
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d_3 (Conv2D)                │ (None, 126, 126, 32)   │           896 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D)   │ (None, 63, 63, 32)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)                │ (None, 61, 61, 64)     │        18,496 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_4 (MaxPooling2D)   │ (None, 30, 30, 64)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)                │ (None, 28, 28, 128)    │        73,856 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_5 (MaxPooling2D)   │ (None, 14, 14, 128)    │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_1 (Flatten)              │ (None, 25088)          │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                  │ (None, 128)            │     3,211,392 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)              │ (None, 128)            │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                  │ (None, 6)              │           774 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 3,305,414 (12.61 MB)
 Trainable params: 3,305,414 (12.61 MB)
```

Frozen Base:



```
    Model Summary(Frozen Base):
••• Model: "sequential_2"
┌───────────────────────────────────┬──────────────────────────┬─────────────────┐
│ Layer (type)                      │ Output Shape             │         Param # │
├───────────────────────────────────┼──────────────────────────┼─────────────────┤
│ mobilenetv2_1.00_224              │ (None, 7, 7, 1280)       │       2,257,984 │
│ (Functional)                      │                          │                 │
├───────────────────────────────────┼──────────────────────────┼─────────────────┤
│ global_average_pooling2d_3        │ (None, 1280)             │               0 │
│ (GlobalAveragePooling2D)          │                          │                 │
├───────────────────────────────────┼──────────────────────────┼─────────────────┤
│ dense_5 (Dense)                   │ (None, 128)              │         163,968 │
├───────────────────────────────────┼──────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)               │ (None, 128)              │               0 │
├───────────────────────────────────┼──────────────────────────┼─────────────────┤
│ dense_6 (Dense)                   │ (None, 6)                │             774 │
└───────────────────────────────────┴──────────────────────────┴─────────────────┘
 Total params: 2,422,726 (9.24 MB)
 Trainable params: 164,742 (643.52 KB)
 Non-trainable params: 2,257,984 (8.61 MB)
Training with frozen base model...
```

Question:

Why does the transfer learning model train faster even though it has more total layers?

Answer:

- Even though MobileNetV2 has more total layers, most layers are frozen (not updated).
- Only the last few layers train, so fewer weights are adjusted every step.
- Fewer trainable parameters → less computation → faster training.

What does "freezing layers" mean and how does it affect training speed?

Answer:

- Setting layer.trainable = False stops updates to those layers.
- Frozen layers act like fixed feature extractors and speed up training.

Explain in your own words: How does using pre-trained weights on ImageNet help with leaf disease classification?

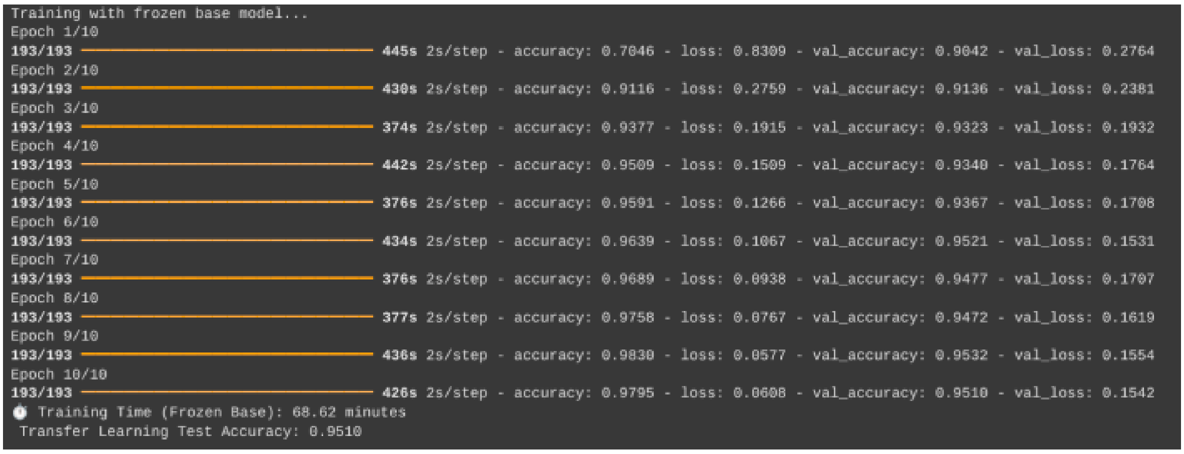
Answer:

- ImageNet has millions of images with common visual patterns (edges, colors, textures).
- These patterns are also useful in plant leaves.
- So, MobileNetV2 already “knows” how to detect such details — we only adjust the top layers for our leaf disease classes.

QUESTION 3

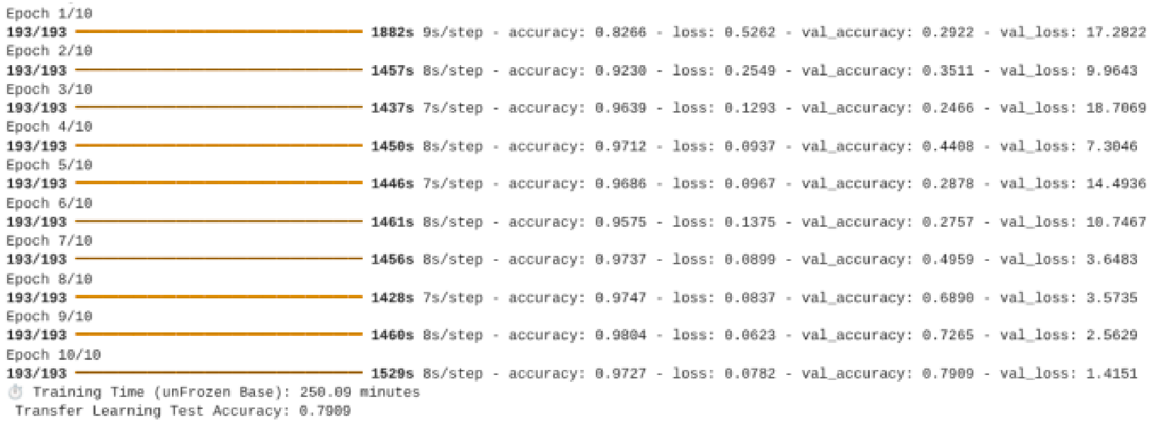
WHAT HAPPENS IF YOU UNFREEZE ALL LAYERS?

Frozen Base: All base_model layers frozen (as shown in assignment code)



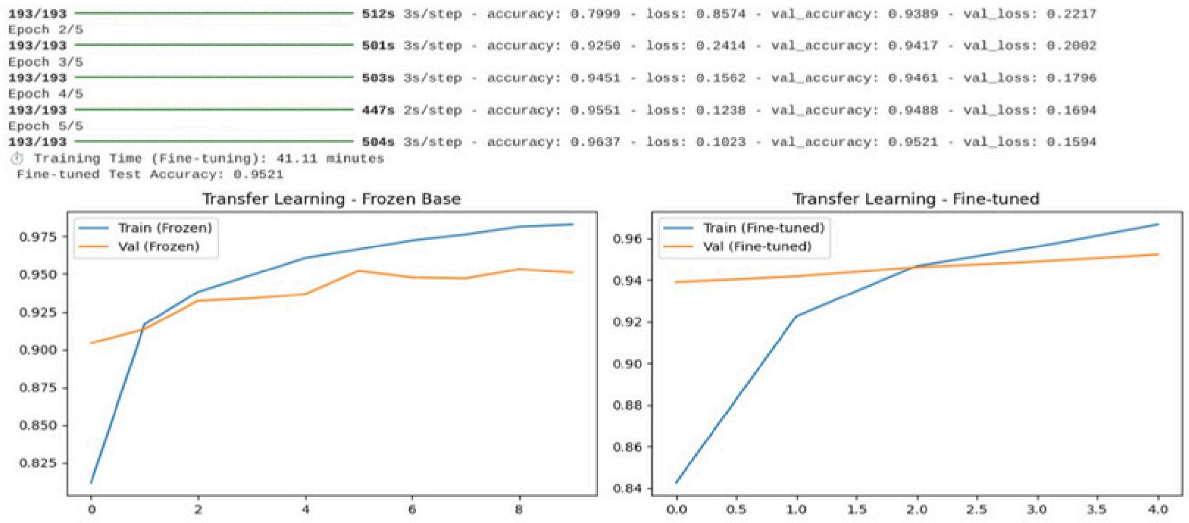
```
Training with frozen base model...
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 445s 2s/step - accuracy: 0.7046 - loss: 0.8309 - val_accuracy: 0.9042 - val_loss: 0.2764
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 430s 2s/step - accuracy: 0.9116 - loss: 0.2759 - val_accuracy: 0.9136 - val_loss: 0.2381
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 374s 2s/step - accuracy: 0.9377 - loss: 0.1915 - val_accuracy: 0.9323 - val_loss: 0.1932
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 442s 2s/step - accuracy: 0.9509 - loss: 0.1509 - val_accuracy: 0.9340 - val_loss: 0.1764
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9591 - loss: 0.1266 - val_accuracy: 0.9367 - val_loss: 0.1708
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 434s 2s/step - accuracy: 0.9639 - loss: 0.1067 - val_accuracy: 0.9521 - val_loss: 0.1531
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9689 - loss: 0.0938 - val_accuracy: 0.9477 - val_loss: 0.1707
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 377s 2s/step - accuracy: 0.9758 - loss: 0.0767 - val_accuracy: 0.9472 - val_loss: 0.1619
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 436s 2s/step - accuracy: 0.9830 - loss: 0.0577 - val_accuracy: 0.9532 - val_loss: 0.1554
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 426s 2s/step - accuracy: 0.9795 - loss: 0.0608 - val_accuracy: 0.9510 - val_loss: 0.1542
⏱ Training Time (Frozen Base): 68.62 minutes
 Transfer Learning Test Accuracy: 0.9510
```

Fully Unfrozen: All layers trainable from the start



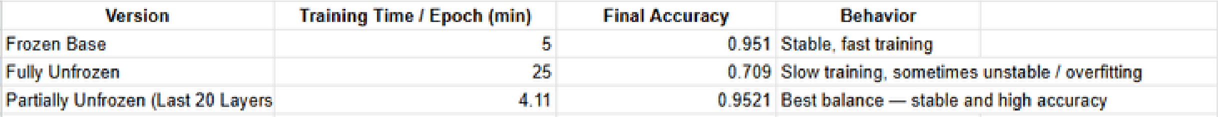
```
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1882s 9s/step - accuracy: 0.8266 - loss: 0.5262 - val_accuracy: 0.2922 - val_loss: 17.2822
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1457s 8s/step - accuracy: 0.9230 - loss: 0.2549 - val_accuracy: 0.3511 - val_loss: 9.9643
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1437s 7s/step - accuracy: 0.9639 - loss: 0.1293 - val_accuracy: 0.2466 - val_loss: 18.7069
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1450s 8s/step - accuracy: 0.9712 - loss: 0.0937 - val_accuracy: 0.4408 - val_loss: 7.3046
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1446s 7s/step - accuracy: 0.9686 - loss: 0.0967 - val_accuracy: 0.2878 - val_loss: 14.4936
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1461s 8s/step - accuracy: 0.9575 - loss: 0.1375 - val_accuracy: 0.2757 - val_loss: 10.7467
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1456s 8s/step - accuracy: 0.9737 - loss: 0.0899 - val_accuracy: 0.4959 - val_loss: 3.6483
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1428s 7s/step - accuracy: 0.9747 - loss: 0.0837 - val_accuracy: 0.6890 - val_loss: 3.5735
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1460s 8s/step - accuracy: 0.9804 - loss: 0.0623 - val_accuracy: 0.7265 - val_loss: 2.5629
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 1529s 8s/step - accuracy: 0.9727 - loss: 0.0782 - val_accuracy: 0.7909 - val_loss: 1.4151
⏱ Training Time (unFrozen Base): 250.09 minutes
 Transfer Learning Test Accuracy: 0.7909
```

Partially Unfrozen: Only last 20 layers trainable (fine-tuning approach)



```
193/193 ━━━━━━━━━━━━━━━━━━━━ 512s 3s/step - accuracy: 0.7999 - loss: 0.8574 - val_accuracy: 0.9389 - val_loss: 0.2217
Epoch 2/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 501s 3s/step - accuracy: 0.9250 - loss: 0.2414 - val_accuracy: 0.9417 - val_loss: 0.2002
Epoch 3/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 503s 3s/step - accuracy: 0.9451 - loss: 0.1562 - val_accuracy: 0.9461 - val_loss: 0.1796
Epoch 4/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 447s 2s/step - accuracy: 0.9551 - loss: 0.1238 - val_accuracy: 0.9488 - val_loss: 0.1694
Epoch 5/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 504s 3s/step - accuracy: 0.9637 - loss: 0.1023 - val_accuracy: 0.9521 - val_loss: 0.1594
⏱ Training Time (Fine-tuning): 41.11 minutes
 Fine-tuned Test Accuracy: 0.9521
```

Version, Accuracy, Training Time & Behaviour.



Version	Training Time / Epoch (min)	Final Accuracy	Behavior	
Frozen Base	5	0.951	Stable, fast training	
Fully Unfrozen	25	0.709	Slow training, sometimes unstable / overfitting	
Partially Unfrozen (Last 20 Layers	4.11	0.9521	Best balance — stable and high accuracy	

Questions to answer:

- **Which approach gives the best test accuracy?**

Answer: Partial fine-tuning (good accuracy & stability).

- **What problems (if any) did you observe when unfreezing all layers from the start?**

Answer: Training slows down and can overfit quickly.

- **Why is fine-tuning (partial unfreezing with low learning rate) often better than unfreezing everything?**

Answer: It keeps strong general ImageNet features and only adjusts the last few layers, saving time and improving stability.