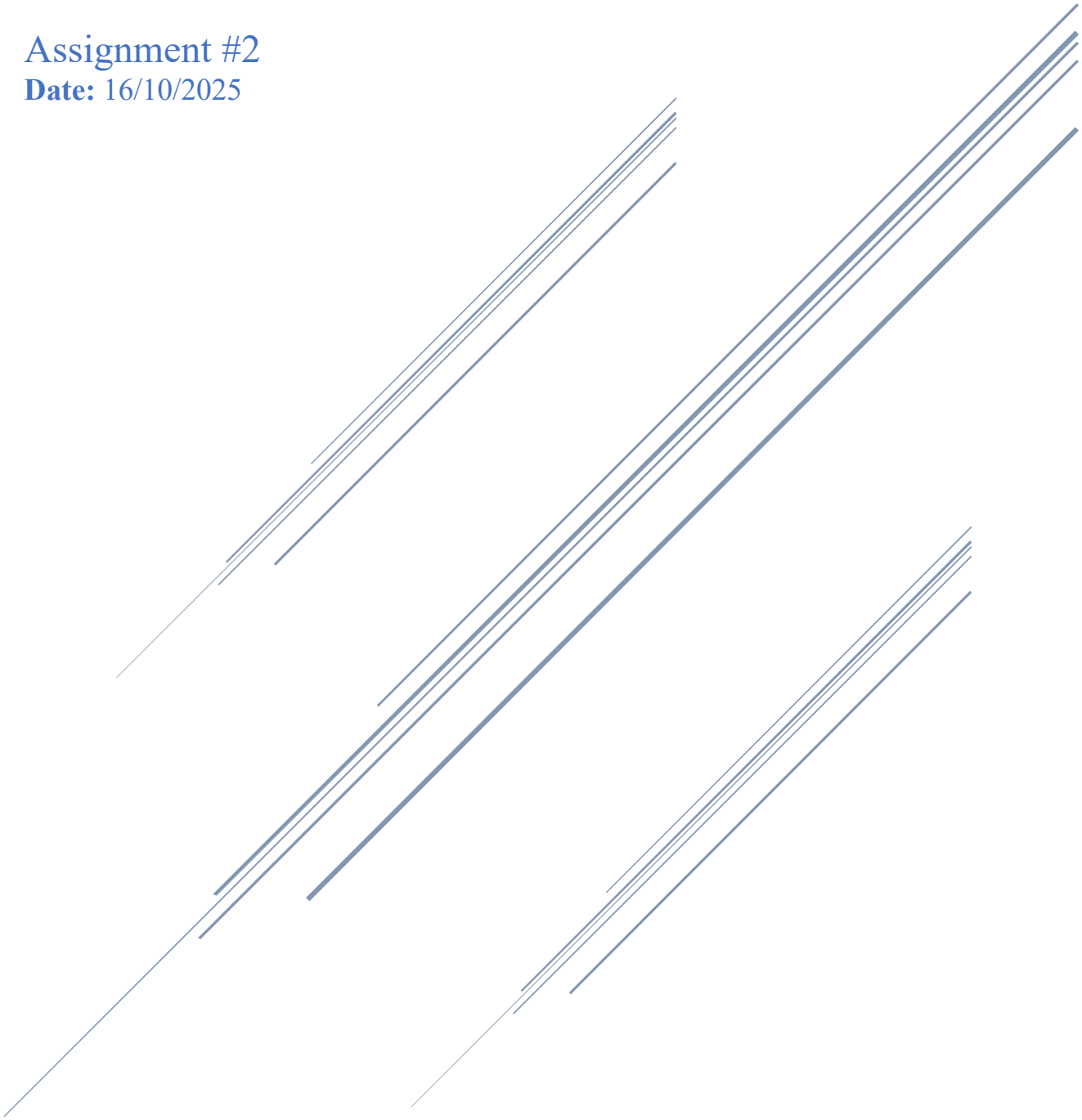
SPECIAL TOPICS IN COMPUTER

Assignment #2

Date: 16/10/2025



From: Syed Hassan Dildar (8984)

To: Ma’am Sadaf Tanvir

Assignment #03

```
import os
import random
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Fix randomness
seed = 42
os.environ['PYTHONHASHSEED'] = str(seed)
random.seed(seed)
np.random.seed(seed)
tf.random.set_seed(seed)

# Load data (same as before)
train_data = tf.keras.utils.image_dataset_from_directory(
   "/content/drive/MyDrive/Train",
    image_size=(224, 224),  # Note: Changed to 224 for MobileNetV2
    batch_size=32,
    label_mode='int',
    shuffle=True,
    seed=seed
)

test_data = tf.keras.utils.image_dataset_from_directory(
    "/content/drive/MyDrive/Test",
    image_size=(224, 224),
    batch_size=32,
    label_mode='int',
    shuffle=False
)

# Get class names before applying map
class_names = train_data.class_names

# Preprocess for MobileNetV2 (values -1 to 1)
train_data = train_data.map(
    lambda x, y: (tf.keras.applications.mobilenet_v2.preprocess_input(x),
y)
)
test_data = test_data.map(
```

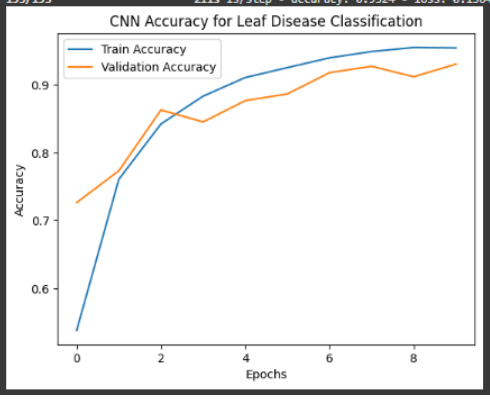
```
    lambda x, y: (tf.keras.applications.mobilenet_v2.preprocess_input(x),
y)
)

# --- TASK 1: Load Pre-trained MobileNetV2 ---
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,  # Remove classification layer
    weights='imagenet'   # Use ImageNet pre-trained weights
)

# Freeze base model (don't train these layers)
base_model.trainable = False

# --- TASK 2: Add Custom Classification Head ---
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3, seed=seed),
    layers.Dense(len(class_names), activation='softmax')
])

# Compile
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# --- TASK 3: Train Model ---
print("Training with frozen base model...")
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=10,
    verbose=1
)

# Evaluate
test_loss, test_acc = model.evaluate(test_data, verbose=0)
print(f" Transfer Learning Test Accuracy: {test_acc:.4f}")

# --- TASK 4 (Optional): Fine-tune top layers ---
print("\n--- Fine-tuning top layers ---")
```

```
base_model.trainable = True

# Freeze all layers except last 20
for layer in base_model.layers[:-20]:
    layer.trainable = False

# Recompile with lower learning rate
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Train again
history_fine = model.fit(
    train_data,
    validation_data=test_data,
    epochs=5,
    verbose=1
)

# Final evaluation
test_loss, test_acc = model.evaluate(test_data, verbose=0)
print(f" Fine-tuned Test Accuracy: {test_acc:.4f}")

# Plot comparison
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train (Frozen)')
plt.plot(history.history['val_accuracy'], label='Val (Frozen)')
plt.title('Transfer Learning - Frozen Base')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_fine.history['accuracy'], label='Train (Fine-tuned)')
plt.plot(history_fine.history['val_accuracy'], label='Val (Fine-tuned)')
plt.title('Transfer Learning - Fine-tuned')
plt.legend()

plt.tight_layout()
plt.show()
```

Question 01 (Tasks)

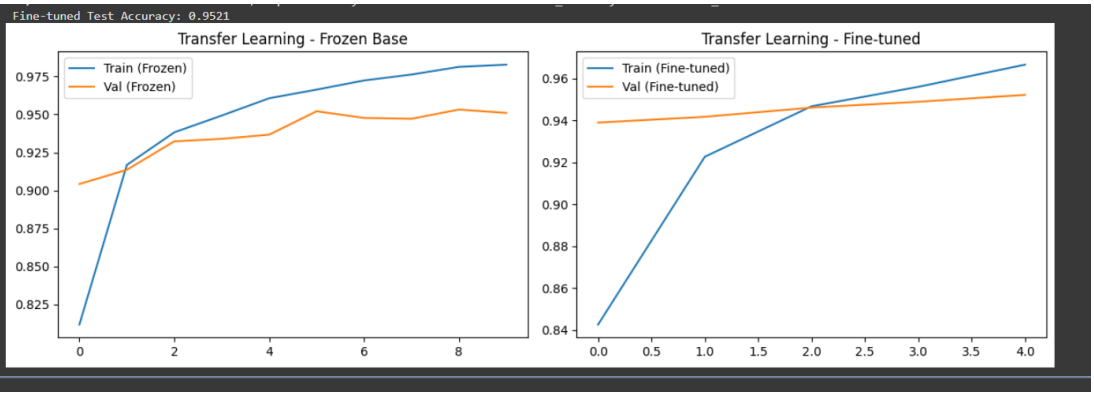
1. CNN Model

```
Found 6152 files belonging to 6 classes.
Found 1817 files belonging to 6 classes.
Epoch 1/10
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
193/193 ━━━━━━━━━━━━━━━━━━━━ 613s 3s/step - accuracy: 0.3932 - loss: 1.4584 - val_accuracy: 0.7265 - val_loss: 0.7965
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 220s 1s/step - accuracy: 0.7326 - loss: 0.7045 - val_accuracy: 0.7733 - val_loss: 0.6466
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 214s 1s/step - accuracy: 0.8249 - loss: 0.5025 - val_accuracy: 0.8630 - val_loss: 0.4022
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 263s 1s/step - accuracy: 0.8651 - loss: 0.3614 - val_accuracy: 0.8453 - val_loss: 0.4325
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 215s 1s/step - accuracy: 0.9052 - loss: 0.2666 - val_accuracy: 0.8767 - val_loss: 0.3640
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 267s 1s/step - accuracy: 0.9185 - loss: 0.2269 - val_accuracy: 0.8866 - val_loss: 0.3438
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 220s 1s/step - accuracy: 0.9314 - loss: 0.1918 - val_accuracy: 0.9180 - val_loss: 0.2527
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 261s 1s/step - accuracy: 0.9447 - loss: 0.1586 - val_accuracy: 0.9274 - val_loss: 0.2393
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 281s 1s/step - accuracy: 0.9571 - loss: 0.1181 - val_accuracy: 0.9119 - val_loss: 0.2984
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 211s 1s/step - accuracy: 0.9524 - loss: 0.1364 - val_accuracy: 0.9307 - val_loss: 0.2361
```



2. MobileNetV2 transfer learning Model

```
Fine-tuned Test Accuracy: 0.9521
```



```
Found 6152 files belonging to 6 classes.
Found 1817 files belonging to 6 classes.
Training with frozen base model...
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 826s 4s/step - accuracy: 0.7046 - loss: 0.8309 - val_accuracy: 0.9042 - val_loss: 0.2764
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 391s 2s/step - accuracy: 0.9116 - loss: 0.2759 - val_accuracy: 0.9136 - val_loss: 0.2381
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 400s 2s/step - accuracy: 0.9377 - loss: 0.1915 - val_accuracy: 0.9323 - val_loss: 0.1932
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 393s 2s/step - accuracy: 0.9509 - loss: 0.1509 - val_accuracy: 0.9340 - val_loss: 0.1764
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 386s 2s/step - accuracy: 0.9591 - loss: 0.1266 - val_accuracy: 0.9367 - val_loss: 0.1708
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 451s 2s/step - accuracy: 0.9639 - loss: 0.1067 - val_accuracy: 0.9521 - val_loss: 0.1531
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 380s 2s/step - accuracy: 0.9689 - loss: 0.0938 - val_accuracy: 0.9477 - val_loss: 0.1707
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 436s 2s/step - accuracy: 0.9758 - loss: 0.0767 - val_accuracy: 0.9472 - val_loss: 0.1619
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 385s 2s/step - accuracy: 0.9830 - loss: 0.0577 - val_accuracy: 0.9532 - val_loss: 0.1554
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 388s 2s/step - accuracy: 0.9795 - loss: 0.0608 - val_accuracy: 0.9510 - val_loss: 0.1542
 Transfer Learning Test Accuracy: 0.9510

--- Fine-tuning top layers ---
Epoch 1/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 476s 2s/step - accuracy: 0.7999 - loss: 0.8574 - val_accuracy: 0.9389 - val_loss: 0.2217
Epoch 2/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 455s 2s/step - accuracy: 0.9250 - loss: 0.2414 - val_accuracy: 0.9417 - val_loss: 0.2002
Epoch 3/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 452s 2s/step - accuracy: 0.9451 - loss: 0.1562 - val_accuracy: 0.9461 - val_loss: 0.1796
Epoch 4/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 503s 2s/step - accuracy: 0.9551 - loss: 0.1238 - val_accuracy: 0.9488 - val_loss: 0.1694
Epoch 5/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 453s 2s/step - accuracy: 0.9637 - loss: 0.1023 - val_accuracy: 0.9521 - val_loss: 0.1594
```

Comparison

Content	CNN	MobileNetV2 transfer
Training Time	35 mins	85 mins
Final test accuracy	0.9062	0.9510

Report:

S.no	Metric	Simple CNN	MobileNetV2 (Frozen Base)
1	**Final Test Accuracy**	0.9062	0.9510
2	**Total Training Time (10 Epochs)**	35 minutes	85 minutes
3	**Trainable Parameters**	3,305,414	202,374 (Classification Head)
4	**Epochs to Reach 80% Accuracy**	2	1

Which model performs better?

MobileNetV2 (transfer learning) performs better with a higher test accuracy of 0.9510, compared to the CNN model's accuracy of 0.9062.

Why does MobileNetV2 perform better?

Because it uses transfer learning, which means:

- It starts with a model that’s already trained on a huge dataset (like ImageNet).

- It has already learned to detect basic things like edges, shapes, and patterns.

- We just fine-tune it to work on our specific task, instead of training everything from zero.

So, it learns faster and better, which gives higher accuracy.

Question 02 (Tasks)

Model Summary() for both models::

📊 Model Summary:
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_7 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_8 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten_2 (Flatten)	(None, 25088)	0
dense_3 (Dense)	(None, 128)	3,211,392
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 6)	774

Total params: 3,305,414 (12.61 MB)
Trainable params: 3,305,414 (12.61 MB)
Non-trainable params: 0 (0.00 B)

```
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 time per epoch for both models:

Training Time	CNN	MobileNetV2
Epoch 1	613 sec	826 sec
Epoch 2	220 sec	391 sec
Epoch 3	214 sec	400 sec
Epoch 4	263 sec	393 sec
Epoch 5	215 sec	386 sec
Epoch 6	267 sec	451 sec
Epoch 7	220 sec	380 sec
Epoch 8	261 sec	436 sec
Epoch 9	281 sec	385 sec
Epoch 10	211 sec	388 sec

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

  The transfer learning model trains faster because most of its layers are pre-trained and frozen, so only a few top layers need to be trained from scratch. This reduces the number of trainable parameters, allowing the model to converge faster.

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

  The transfer learning model trains faster because most of its layers are pre-trained and frozen, so only a few top layers need to be trained from scratch. This reduces the number of trainable parameters, allowing the model to converge faster.

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

  Pre-trained weights from ImageNet help leaf disease classification by providing a foundation of general image features, allowing the model to focus on learning disease-specific features.

Question 03 (Tasks)

1. **Frozen Base:**

```
Training with frozen base model...
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 445s 2s/step - accuracy: 0.7046 - loss: 0.8309 -
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 430s 2s/step - accuracy: 0.9116 - loss: 0.2759 -
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 374s 2s/step - accuracy: 0.9377 - loss: 0.1915 -
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 442s 2s/step - accuracy: 0.9509 - loss: 0.1509 -
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9591 - loss: 0.1266 -
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434s 2s/step - accuracy: 0.9639 - loss: 0.1067 -
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.9689 - loss: 0.0938 -
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 377s 2s/step - accuracy: 0.9758 - loss: 0.0767 -
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 436s 2s/step - accuracy: 0.9830 - loss: 0.0577 -
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 426s 2s/step - accuracy: 0.9795 - loss: 0.0608 -
⏱ Training Time (Frozen Base): 68.62 minutes
 Transfer Learning Test Accuracy: 0.9510
```

2. Fully Unfrozen:

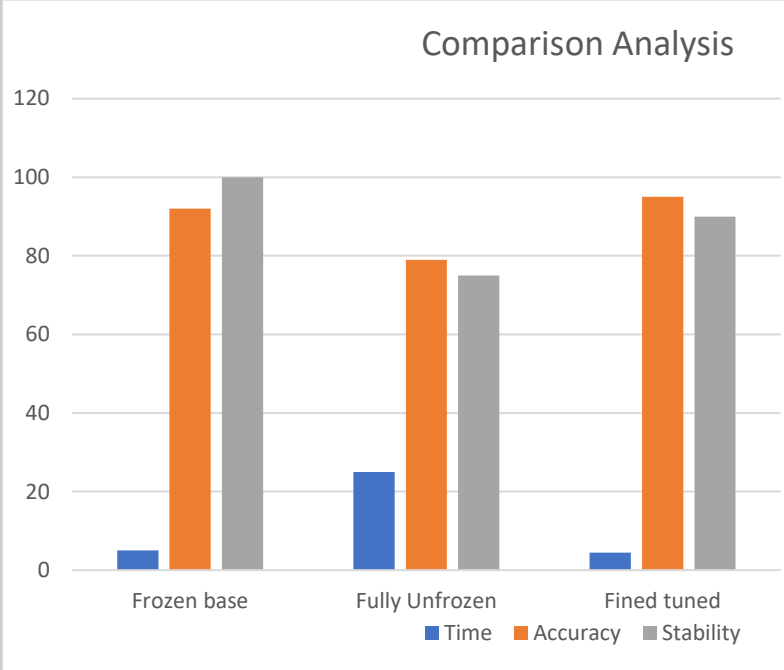
```
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1882s 9s/step - accuracy: 0.8266 - loss: 0.5262 -
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1457s 8s/step - accuracy: 0.9230 - loss: 0.2549 -
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1437s 7s/step - accuracy: 0.9639 - loss: 0.1293 -
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1450s 8s/step - accuracy: 0.9712 - loss: 0.0937 -
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1446s 7s/step - accuracy: 0.9686 - loss: 0.0967 -
Epoch 6/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1461s 8s/step - accuracy: 0.9575 - loss: 0.1375 -
Epoch 7/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1456s 8s/step - accuracy: 0.9737 - loss: 0.0899 -
Epoch 8/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1428s 7s/step - accuracy: 0.9747 - loss: 0.0837 -
Epoch 9/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1460s 8s/step - accuracy: 0.9804 - loss: 0.0623 -
Epoch 10/10
193/193 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1529s 8s/step - accuracy: 0.9727 - loss: 0.0782 -
⏱ Training Time (unFrozen Base): 250.09 minutes
 Transfer Learning Test Accuracy: 0.7909
```

3. Fine Tuned:

```
--- Fine-tuning top layers ---
Epoch 1/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 476s 2s/step - accuracy: 0.7999 - loss: 0.8574 -
Epoch 2/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 455s 2s/step - accuracy: 0.9250 - loss: 0.2414 -
Epoch 3/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 452s 2s/step - accuracy: 0.9451 - loss: 0.1562 -
Epoch 4/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 503s 2s/step - accuracy: 0.9551 - loss: 0.1238 -
Epoch 5/5
193/193 ━━━━━━━━━━━━━━━━━━━━ 453s 2s/step - accuracy: 0.9637 - loss: 0.1023 -
 Fine-tuned Test Accuracy: 0.9521
```

Report:

Content	Frozen Base	Fully Unfrozen	Fine Tuned
Training Time per epoch	3-5 mins	22-25 mins	3.5-4.5 mins
Final test accuracy	0.9510	0.7909	0.9521
Behavior	Stable, no overfitting	Unstable, some overfitting	Stable, minimal overfitting



	Frozen Base	Fully Unfrozen	Fine Tuned
Training Time per epoch	3-5 mins	22-25 mins	3.5-4.5 mins
Final test accuracy	0.9510	0.7909	0.9521
Behavior	Stable, no overfitting	Unstable, some overfitting	Stable, minimal overfitting

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

  Fine Tuned gives the best test accuracy with 0.9521, closely followed by Frozen Base with 0.9510.

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

  - **Unstable training:** Unfreezing all layers from the start led to unstable training behavior.
  - **Overfitting:** The model showed signs of overfitting, likely due to the large number of trainable parameters and insufficient regularization.

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

  1. Reduces overfitting
  2. Preserves pre-trained knowledge
  3. Promotes stable training

  This approach helps adapt pre-trained models to specific tasks while avoiding overfitting