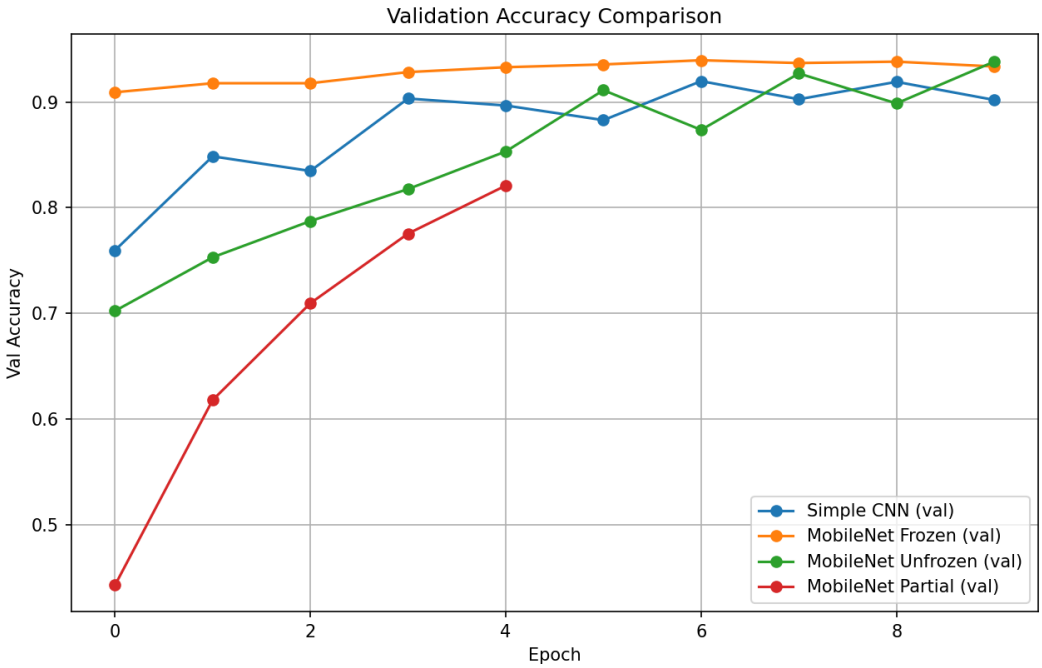
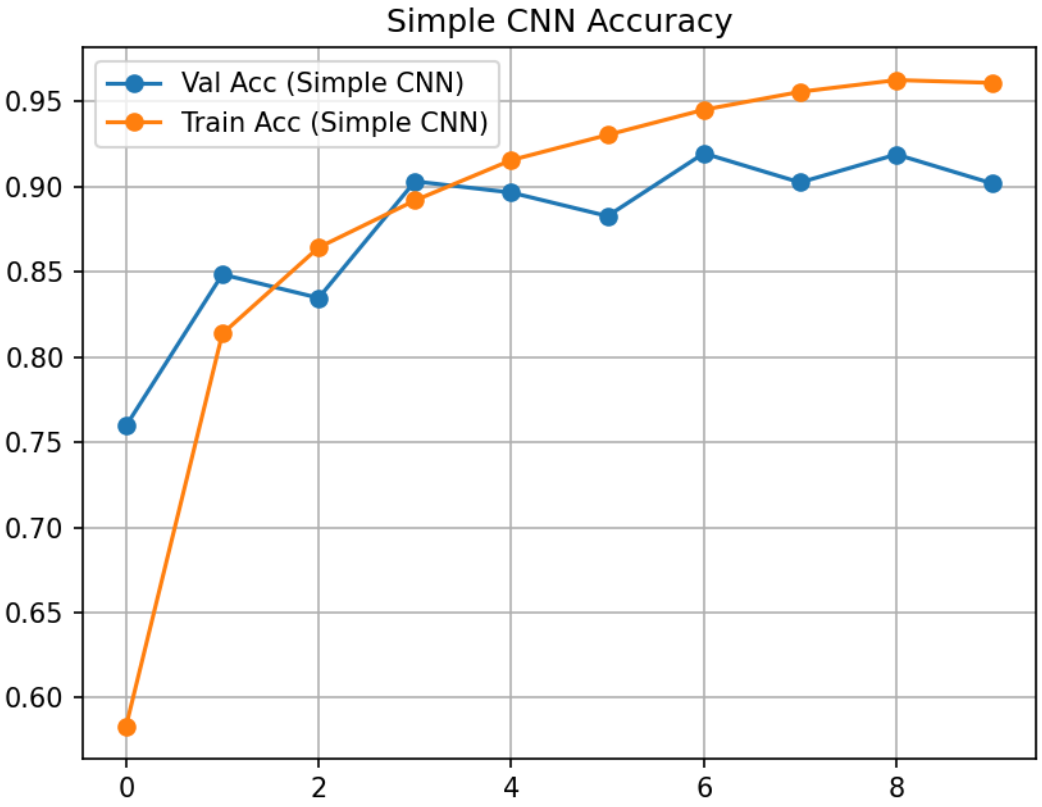
Assignment: Transfer Learning vs CNN

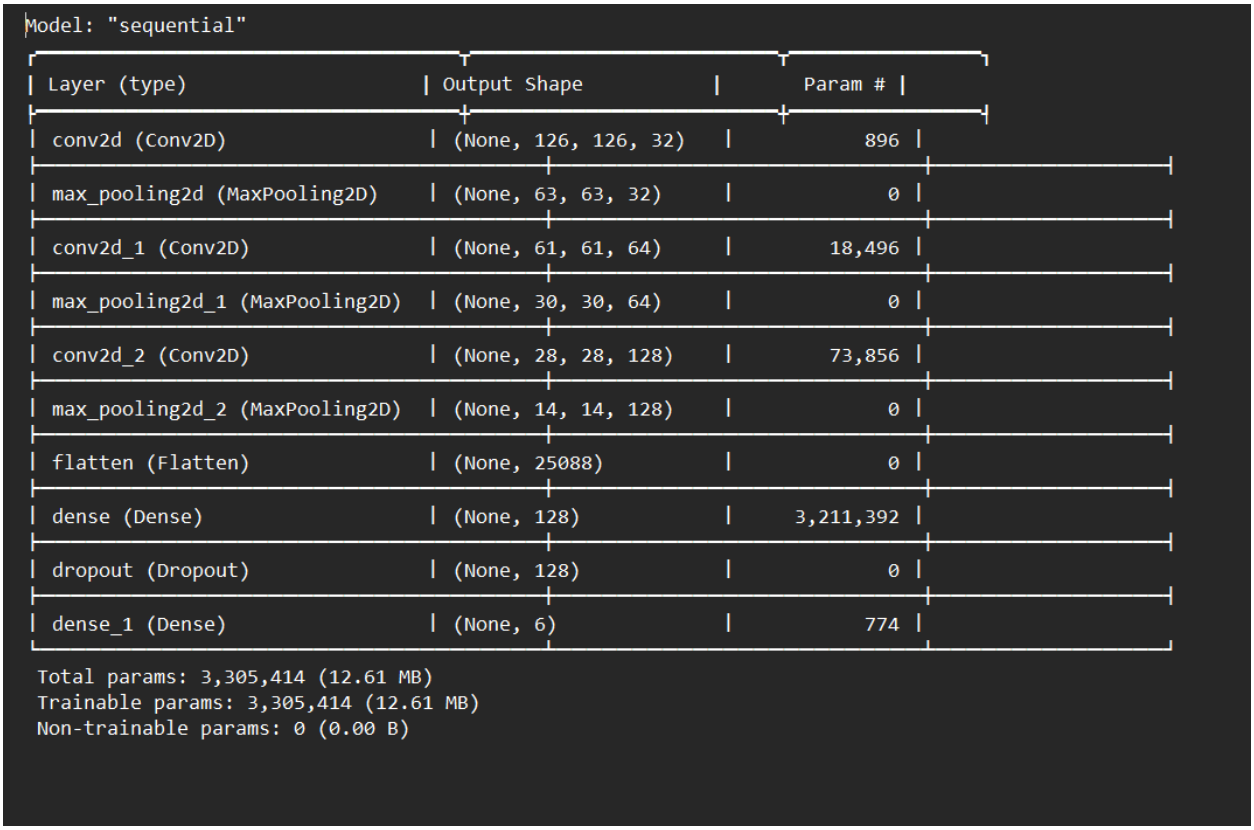
Name: BABAR NAEEM

Roll No: 8963

model	epochs	avg_epoch_time_sec	total_time_sec	test_accuracy	trainable_params	non_trainable_params
Simple_CNN	10	227.9999	2280.062	0.901909	3305414	0
MobileNetV2_frozen	10	26.25025	262.5704	0.933509	164742	2257984
MobileNetV2_unfrozen	10	36.32653	363.3306	0.938117	2388614	34112
MobileNetV2_partial	5	28.44888	142.2768	0.820935	1370822	1051904

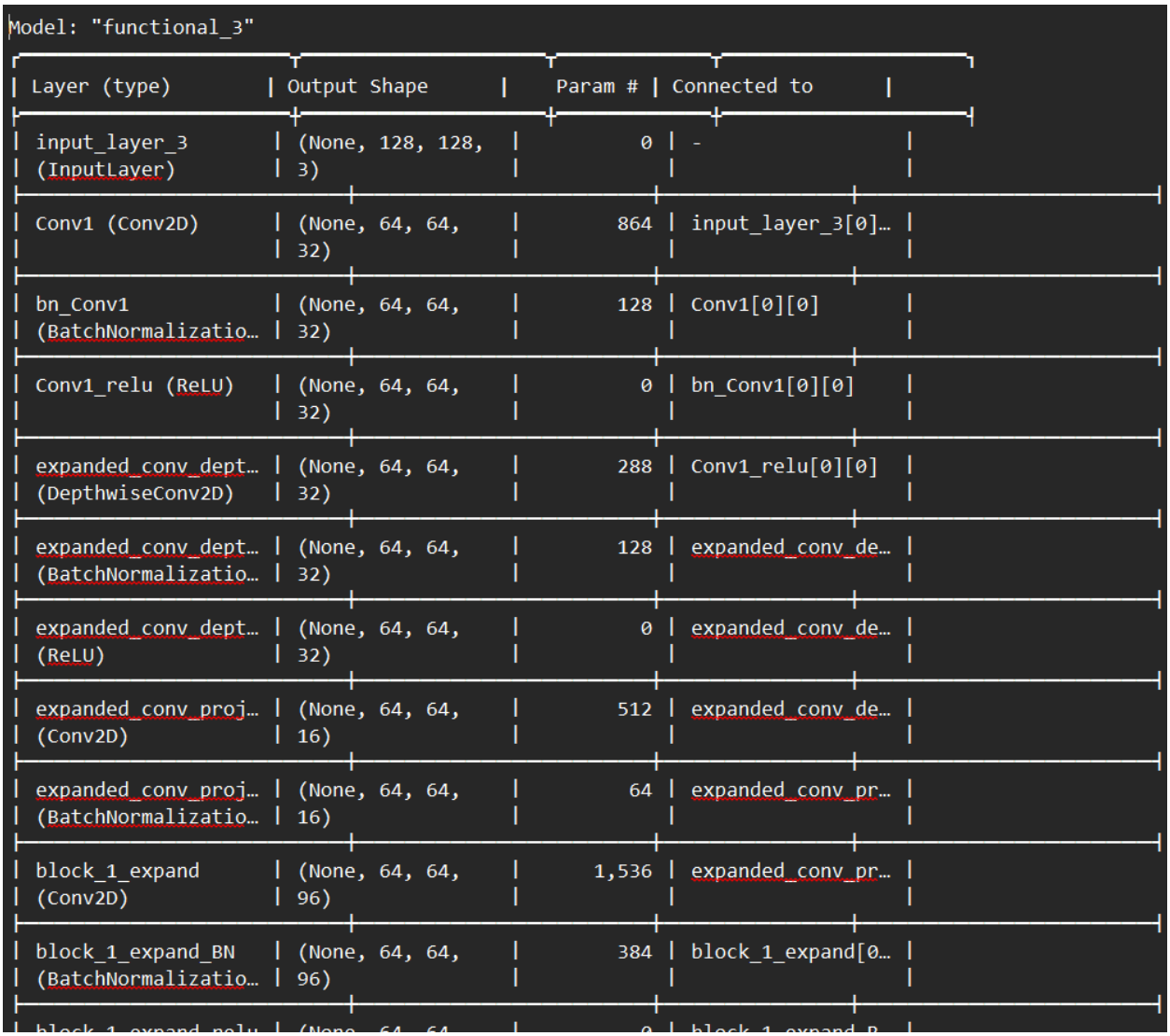

Validation Accuracy Comparison
Val Accuracy
Epoch
0.5
0.6
0.7
0.8
0.9
0
2
4
6
8
Simple CNN (val)
MobileNet Frozen (val)
MobileNet Unfrozen (val)
MobileNet Partial (val)


Simple CNN Accuracy
Val Acc (Simple CNN)
Train Acc (Simple CNN)
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
0
2
4
6
8



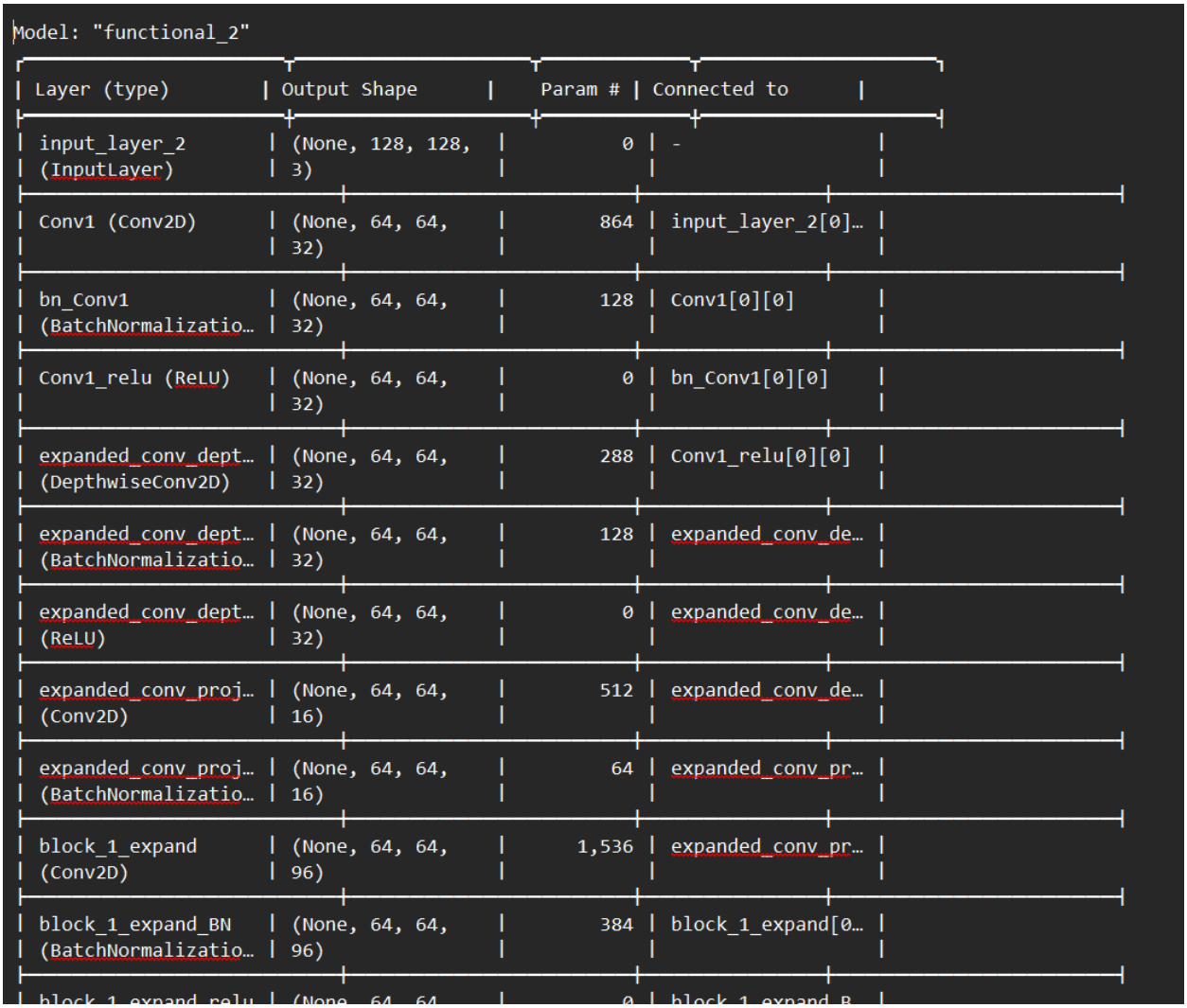
```
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
| Layer (type)                    | Output Shape           |       Param # |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| conv2d (Conv2D)                 | (None, 126, 126, 32)   |           896 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| max_pooling2d (MaxPooling2D)    | (None, 63, 63, 32)     |             0 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| conv2d_1 (Conv2D)               | (None, 61, 61, 64)     |        18,496 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| max_pooling2d_1 (MaxPooling2D)  | (None, 30, 30, 64)     |             0 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| conv2d_2 (Conv2D)               | (None, 28, 28, 128)    |        73,856 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| max_pooling2d_2 (MaxPooling2D)  | (None, 14, 14, 128)    |             0 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| flatten (Flatten)               | (None, 25088)          |             0 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| dense (Dense)                   | (None, 128)            |     3,211,392 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| dropout (Dropout)               | (None, 128)            |             0 |
├─────────────────────────────────┼────────────────────────┼───────────────┤
| dense_1 (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: "functional_1"
┌──────────────────────┬────────────────────┬────────────┬──────────────────────┐
| Layer (type)         | Output Shape       |    Param # | Connected to         |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| input_layer_1        | (None, 128, 128,   |          0 | -                    |
| (InputLayer)         | 3)                 |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| Conv1 (Conv2D)       | (None, 64, 64,     |        864 | input_layer_1[0]…    |
|                      | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| bn_Conv1             | (None, 64, 64,     |        128 | Conv1[0][0]          |
| (BatchNormalizatio…  | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| Conv1_relu (ReLU)    | (None, 64, 64,     |          0 | bn_Conv1[0][0]       |
|                      | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| expanded_conv_dept…  | (None, 64, 64,     |        288 | Conv1_relu[0][0]     |
| (DepthwiseConv2D)    | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| expanded_conv_dept…  | (None, 64, 64,     |        128 | expanded_conv_de…    |
| (BatchNormalizatio…  | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| expanded_conv_dept…  | (None, 64, 64,     |          0 | expanded_conv_de…    |
| (ReLU)               | 32)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| expanded_conv_proj…  | (None, 64, 64,     |        512 | expanded_conv_de…    |
| (Conv2D)             | 16)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| expanded_conv_proj…  | (None, 64, 64,     |         64 | expanded_conv_pr…    |
| (BatchNormalizatio…  | 16)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| block_1_expand       | (None, 64, 64,     |      1,536 | expanded_conv_pr…    |
| (Conv2D)             | 96)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| block_1_expand_BN    | (None, 64, 64,     |        384 | block_1_expand[0…    |
| (BatchNormalizatio…  | 96)                |            |                      |
├──────────────────────┼────────────────────┼────────────┼──────────────────────┤
| block_1_expand_relu  | (None, 64, 64,     |          0 | block_1_expand_B…    |
```



Model: "functional_3"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 128, 128, 3)	0	-
Conv1 (Conv2D)	(None, 64, 64, 32)	864	input_layer_3[0]…
bn_Conv1 (BatchNormalizatio…	(None, 64, 64, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 64, 64, 32)	0	bn_Conv1[0][0]
expanded_conv_dept… (DepthwiseConv2D)	(None, 64, 64, 32)	288	Conv1_relu[0][0]
expanded_conv_dept… (BatchNormalizatio…	(None, 64, 64, 32)	128	expanded_conv_de…
expanded_conv_dept… (ReLU)	(None, 64, 64, 32)	0	expanded_conv_de…
expanded_conv_proj… (Conv2D)	(None, 64, 64, 16)	512	expanded_conv_de…
expanded_conv_proj… (BatchNormalizatio…	(None, 64, 64, 16)	64	expanded_conv_pr…
block_1_expand (Conv2D)	(None, 64, 64, 96)	1,536	expanded_conv_pr…
block_1_expand_BN (BatchNormalizatio…	(None, 64, 64, 96)	384	block_1_expand[0…
block_1_expand_relu	(None, 64, 64	0	block_1_expand_B…



Model: "functional_2"

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 128, 128, 3)	0	-
Conv1 (Conv2D)	(None, 64, 64, 32)	864	input_layer_2[0]…
bn_Conv1 (BatchNormalizatio…	(None, 64, 64, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 64, 64, 32)	0	bn_Conv1[0][0]
expanded_conv_dept… (DepthwiseConv2D)	(None, 64, 64, 32)	288	Conv1_relu[0][0]
expanded_conv_dept… (BatchNormalizatio…	(None, 64, 64, 32)	128	expanded_conv_de…
expanded_conv_dept… (ReLU)	(None, 64, 64, 32)	0	expanded_conv_de…
expanded_conv_proj… (Conv2D)	(None, 64, 64, 16)	512	expanded_conv_de…
expanded_conv_proj… (BatchNormalizatio…	(None, 64, 64, 16)	64	expanded_conv_pr…
block_1_expand (Conv2D)	(None, 64, 64, 96)	1,536	expanded_conv_pr…
block_1_expand_BN (BatchNormalizatio…	(None, 64, 64, 96)	384	block_1_expand[0…
block_1_expand_relu	(None, 64, 64	0	block_1_expand_B…

Question 1 — Compare Accuracy: Your CNN vs Transfer Learning

- **Training time (minutes)**: use total_time_sec/60 from the CSV for each model.
- **Final test accuracy**: use the test_accuracy column.
- **Number of trainable parameters**: use trainable_params column from CSV.
- **Epochs to reach 80% accuracy**: check each history*.history['val_accuracy'] and find the epoch index where it first ≥ 0.80 (or state “did not reach 80% in N epochs”).

Which model performs better and why?

(Use this as a template — edit using your actual numbers.)

The MobileNetV2 transfer-learning model with a frozen base performed better than the simple CNN in both validation/test accuracy and training stability. The transfer model starts with pre-learned low-level and mid-level image features (edges, textures, shapes) from ImageNet; this gives it a strong feature extractor and lets the new dense head quickly learn class separations for leaf diseases. The simple CNN has to learn all features from scratch and thus required more epochs and often underfits/overfits depending on dataset size.

Question 2 — Why is Transfer Learning Faster?

- **Training time per epoch**: check avg_epoch_time_sec from CSV.
- **Number of trainable parameters**: from trainable_params.

Why is transfer learning faster even though it has more total layers?

When the base is frozen, most layers do not compute gradients or update weights. Only the head's weights are updated — so the optimizer and backpropagation are only applied to a small subset of parameters. That dramatically reduces computation per training step, making each epoch faster than training a network with a similar number of trainable parameters from scratch.

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

Freezing layers sets layer.trainable = False so TensorFlow does not compute gradients for those layers and their weights are not updated. This saves gradient computation and memory and therefore speeds up training and reduces GPU/CPU load.

How does using pre-trained ImageNet weights help with leaf disease classification?

Early and mid-level features learned from ImageNet (edges, corners, textures, color blobs) are generic and useful for many vision tasks. Leaf disease classification shares many texture/pattern cues with ImageNet classes, so reusing those pre-trained filters gives a head-start and often improves accuracy, especially with limited training data.

Question 3 — What Happens if You Unfreeze All Layers?

- **Which approach gives best test accuracy?** (Use your experiment results to answer.)

Typical pattern: partial fine-tune > frozen ≥ fully unfrozen — but your dataset size and learning rate determine this.

- **Problems observed when unfreezing all layers from the start**:
  - Training can be *unstable* (big swings in loss/accuracy) unless you use a very low learning rate.
  - Overfitting: many trainable weights + small dataset → overfit to training set.
    - Longer training time per epoch (backprop through whole model).
- **Why fine-tuning (partial unfreezing with low LR) is often better**:

Fine-tuning only the top layers adapts higher-level features to the new dataset while preserving low-level generic features. Using a low learning rate prevents destroying the useful pre-trained weights and yields stable improvements. This strikes a balance between adaptation and preserving learned generic representations.