Question/Answers

Answer 1: Transfer Learning involves taking models designed for initial tasks and applying them as foundational models for subsequent tasks.

Why it's used:

- A pre-trained model such as MobileNetV2 which benefits from training on the extensive ImageNet dataset saves us from the time-consuming and data-intensive process of starting from scratch.
- These pre-trained models have already learned useful feature representations (like edges, textures, shapes) that work well for many visual tasks like object detection and classification.

Key Difference:

From Scratch training initializes weights randomly while demanding a large dataset and extended training duration whereas transfer learning begins with pre-trained weights instead of random initializations while retraining only the top layers or specific layers on your smaller dataset.

Answer 2: Fine-tuning involves training selected top layers of the pre-trained model together with the new classifier layers while keeping them unfrozen.

Why it's useful:

- The feature extraction process involves training only the newly added classification layers while keeping the base model layers static.
- During fine-tuning the base model undergoes training of its deeper layers using a small learning rate to personalize pre-existing features for your specific dataset (such as Cats vs Dogs).

Answer 3: The base model's pre-trained weights remain unchanged when freezing is applied.

Reasons:

- The first convolutional layers captured universal ImageNet patterns such as edges and textures and shapes.
- The learned features of pre-trained models may deteriorate if you retrain them prematurely when working with limited datasets.
- Freezing maintains the stability of these features during the new classifier training process.

Answer 4: Data Augmentation artificially increases your dataset size through the generation of modified image versions including rotations and flips.

Why important:

- The technique prevents overfitting while simultaneously improving the model's ability to generalize.
- The model becomes resistant to slight changes and modifications in input images through this method.

Answer 5: Screenshot of Pre-trained MobileNetV2 Loaded (without top layers):

```
# Load pre-trained MobileNetV2 without top layers (Transfer Learning Step)
base_model = MobileNetV2(input_shape=(160, 160, 3),
                         include_top=False,
                         weights='imagenet')
```

Answer 6: Screenshot Where Pre-trained Model is Set to Non-Trainable:

```
                         weights='imagenet')
base_model.trainable = False  # Freeze the base
```

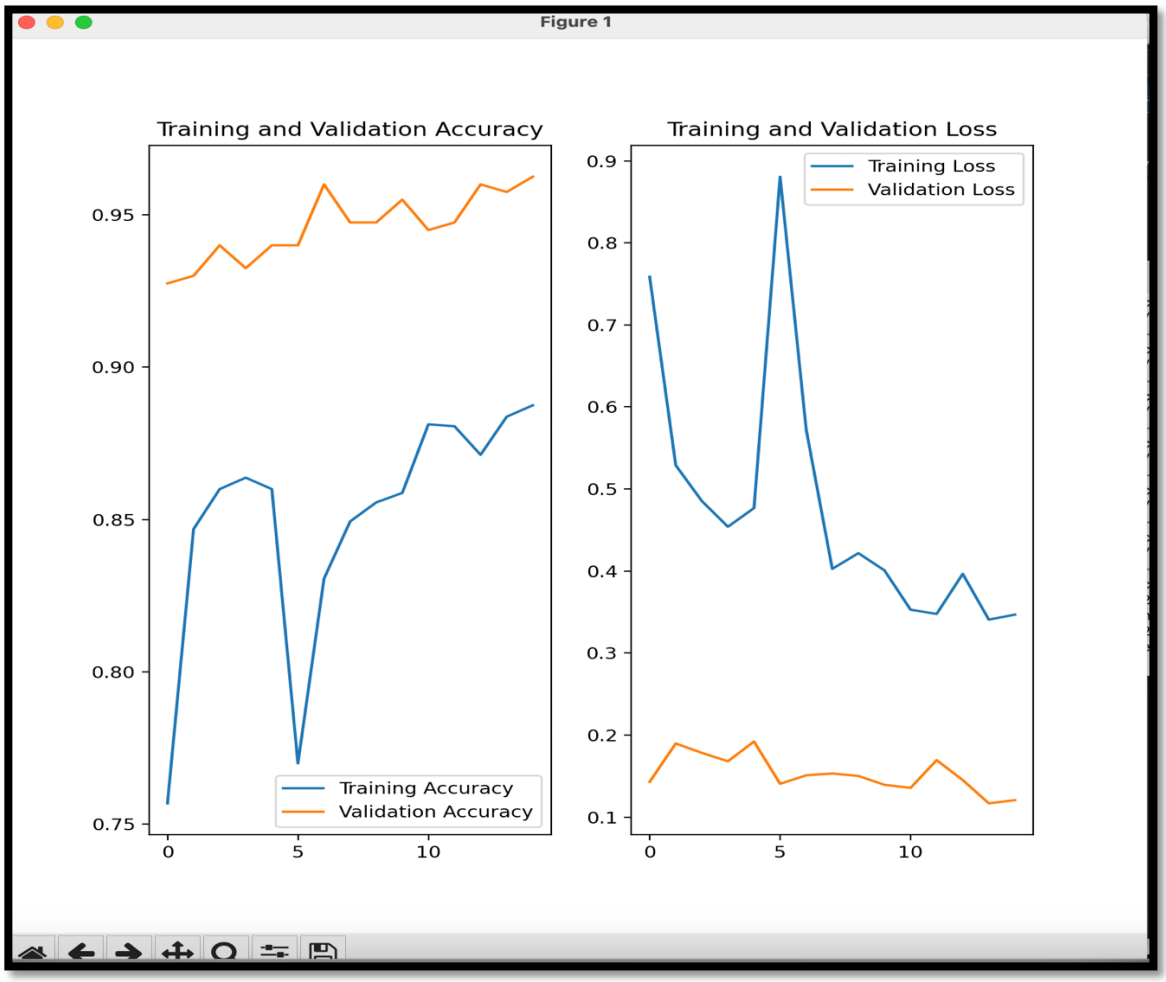
Answer 7: Screenshot of Data Augmentation Layers:

```
# Data augmentation layers
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip('horizontal'),
    layers.RandomRotation(0.2),
])
```

Answer 8: Screenshot of New Classifier Layers Added on Top of the Base Model:

```
60
61  # Add new classifier on top of MobileNetV2
62  model = models.Sequential([
63      data_augmentation,
64      base_model,
65      layers.GlobalAveragePooling2D(),
66      layers.Dense(128, activation='relu'),
67      layers.Dropout(0.5),
68      layers.Dense(1, activation='sigmoid')
69  ])
70
```

Outputs:




```
  self._warn_if_super_not_called()
Epoch 1/5
2025-04-24 10:23:02.942551: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:117] Plugin optimi
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 186ms/step - accuracy: 0.4732 - loss: 0.9552/Users/hritikanand/Library/CloudStorage/OneDr
/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__
sing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 270ms/step - accuracy: 0.4734 - loss: 0.9546 - val_accuracy: 0.5000 - val_loss: 0.7181
Epoch 2/5
50/50 ━━━━━━━━━━━━━━━━━━━━ 12s 235ms/step - accuracy: 0.5043 - loss: 0.8595 - val_accuracy: 0.5000 - val_loss: 0.6922
Epoch 3/5
50/50 ━━━━━━━━━━━━━━━━━━━━ 12s 230ms/step - accuracy: 0.4824 - loss: 0.9111 - val_accuracy: 0.5600 - val_loss: 0.6926
Epoch 4/5
50/50 ━━━━━━━━━━━━━━━━━━━━ 12s 231ms/step - accuracy: 0.4718 - loss: 0.8751 - val_accuracy: 0.5000 - val_loss: 0.6968
Epoch 5/5
50/50 ━━━━━━━━━━━━━━━━━━━━ 12s 230ms/step - accuracy: 0.5131 - loss: 0.8160 - val_accuracy: 0.5000 - val_loss: 0.6950
Epoch 6/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 28s 379ms/step - accuracy: 0.4763 - loss: 0.8662 - val_accuracy: 0.5000 - val_loss: 0.6964
Epoch 7/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 333ms/step - accuracy: 0.5110 - loss: 0.8144 - val_accuracy: 0.5000 - val_loss: 0.7333
Epoch 8/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 349ms/step - accuracy: 0.5019 - loss: 0.8246 - val_accuracy: 0.5000 - val_loss: 0.7466
Epoch 9/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 339ms/step - accuracy: 0.5464 - loss: 0.7986 - val_accuracy: 0.5000 - val_loss: 0.7579
Epoch 10/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 346ms/step - accuracy: 0.5281 - loss: 0.8091 - val_accuracy: 0.5000 - val_loss: 0.7816
2025-04-24 10:25:42.903 python[2963:145703] +[IMKClient subclass]: chose IMKClient_Modern
2025-04-24 10:25:42.903 python[2963:145703] +[IMKInputSession subclass]: chose IMKInputSession_Modern
(venv) (base) hritikanand@Hritiks-MacBook-Air Lab 06 % python transfer.py
✅ GPU is available and configured!
Found 1600 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
2025-04-24 10:33:16.044609: I metal_plugin/src/device/metal_device.cc:1154] Metal device set to: Apple M3
2025-04-24 10:33:16.044661: I metal_plugin/src/device/metal_device.cc:296] systemMemory: 16.00 GB
2025-04-24 10:33:16.044673: I metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 5.33 GB
2025-04-24 10:33:16.044694: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:305] Could no
 built with NUMA support.
```