```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os

# Defining the path to the provided ZIP file in my Google Drive
zip_path = '/content/drive/My Drive/Colab Notebooks/cats_vs_dogs_smal

# Defining extracted files path into My Google Drive Folder
extract_path = "/content/drive/My Drive/Colab Notebooks/"

# Unzipping the files quietely
!unzip -q "{zip_path}" -d "{extract_path}"

print("Files unzipped successfully")
```

```
Files unzipped successfully
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import shutil
import pathlib
import matplotlib.pyplot as plt

# Defining Original Base Directory
original_base_dir = pathlib.Path("/content/drive/My Drive/Colab Noteb
# Defining Original Base Subdirectories
original_train_dir = original_base_dir / "train"
original_validation_dir = original_base_dir / "validation"
original_test_dir = original_base_dir / "test"

# Dictionary to store all the results
results = {}
```

```
# Defining Subsets
def make_subset(subset_name, train_size, validation_size, test_size):
    new_base_dir = pathlib.Path(f"/content/{subset_name}")
    if new_base_dir.exists():
        shutil.rmtree(new_base_dir)

    print(f"Creating new dataset subset: {new_base_dir}")

    # Helper function to copy files
    def copy_files(src_dir, dst_dir, num_files):
```

```
        os.makedirs(dst_dir, exist_ok=True)
        # List all files in the source directory
        all_files = sorted([f for f in os.listdir(src_dir) if os.path
        # Copy the first 'num_files' from the list
        files_to_copy = all_files[:num_files]
        for fname in files_to_copy:
            src_file = src_dir / fname
            shutil.copyfile(src_file, dst_dir / fname)
        print(f"Copied {len(files_to_copy)} files from {src_dir} to {


    # --- Creating Training Set ---
    copy_files(original_train_dir / "cats", new_base_dir / "train" /
    copy_files(original_train_dir / "dogs", new_base_dir / "train" /

    # --- Creating Validation Set ---
    copy_files(original_validation_dir / "cats", new_base_dir / "vali
    copy_files(original_validation_dir / "dogs", new_base_dir / "vali

    # --- Creating Test Set ---
    copy_files(original_test_dir / "cats", new_base_dir / "test" / "ca
    copy_files(original_test_dir / "dogs", new_base_dir / "test" / "do


    return new_base_dir
```

```
# Loading Datasets
def get_datasets(base_dir, image_size=(180, 180), batch_size=32):
    train_dataset = keras.utils.image_dataset_from_directory(
        base_dir / "train",
        image_size=image_size,
        batch_size=batch_size
    )
    validation_dataset = keras.utils.image_dataset_from_directory(
        base_dir / "validation",
        image_size=image_size,
        batch_size=batch_size
    )
    test_dataset = keras.utils.image_dataset_from_directory(
        base_dir / "test",
        image_size=image_size,
        batch_size=batch_size
    )
    return train_dataset, validation_dataset, test_dataset

def plot_history(history, title):
    acc = history.history["accuracy"]
    val_acc = history.history["val_accuracy"]
    loss = history.history["loss"]
    val_loss = history.history["val_loss"]
```

```
    epochs = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 4))
    plt.suptitle(title, y=1.02)

    plt.subplot(1, 2, 1)
    plt.plot(epochs, acc, "bo", label="Training acc")
    plt.plot(epochs, val_acc, "b", label="Validation acc")
    plt.title("Training and validation accuracy")
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs, loss, "bo", label="Training loss")
    plt.plot(epochs, val_loss, "b", label="Validation loss")
    plt.title("Training and validation loss")
    plt.legend()

    plt.show()
```

```
# Q1,2,3
# Creating Subsets

# Step 1 subset: Train=1000, Val=500, Test=500
subset_1_dir = make_subset("subset_1000_500_500",
                           train_size=1000,
                           validation_size=500,
                           test_size=500)

# Step 2 subset: Train=2000, Val=500, Test=500
subset_2_dir = make_subset("subset_2000_500_500",
                           train_size=2000,
                           validation_size=500,
                           test_size=500)

# Step 3 subset: "Ideal" size (using full original dataset)
subset_3_dir = make_subset("subset_2000_1000_1000",
                           train_size=2000,
                           validation_size=1000,
                           test_size=1000)

#  Loading new subsetted datasets
print("\nLoading datasets...")
train_ds_1, val_ds_1, test_ds_1 = get_datasets(subset_1_dir)
train_ds_2, val_ds_2, test_ds_2 = get_datasets(subset_2_dir)
train_ds_3, val_ds_3, test_ds_3 = get_datasets(subset_3_dir)
```

```
Creating new dataset subset: /content/subset_1000_500_500
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
```

```
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Creating new dataset subset: /content/subset_2000_500_500
Copied 1000 files from /content/drive/My Drive/Colab Notebooks/cats_vs
Copied 1000 files from /content/drive/My Drive/Colab Notebooks/cats_vs
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 250 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Creating new dataset subset: /content/subset_2000_1000_1000
Copied 1000 files from /content/drive/My Drive/Colab Notebooks/cats_vs
Copied 1000 files from /content/drive/My Drive/Colab Notebooks/cats_vs
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_
Copied 500 files from /content/drive/My Drive/Colab Notebooks/cats_vs_

Loading datasets...
Found 1000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 2000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
# Building Scratch Model
def build_model_from_scratch():

    # Data Augmentation layers
    data_augmentation = keras.Sequential(
        [
            layers.RandomFlip("horizontal"),
            layers.RandomRotation(0.1),
            layers.RandomZoom(0.2),
        ]
    )

    # Model Architecture
    inputs = keras.Input(shape=(180, 180, 3))
    x = data_augmentation(inputs)

    x = layers.Rescaling(1./255)(x)
    x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(
```

```
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(:
    x = layers.Flatten()(x)

    # Dropout for regularization
    x = layers.Dropout(0.5)(x)

    outputs = layers.Dense(1, activation="sigmoid")(x)
    model = keras.Model(inputs=inputs, outputs=outputs)

    # Compiling the model
    model.compile(loss="binary_crossentropy",
                  optimizer="rmsprop",
                  metrics=["accuracy"])

    return model
```

```
def train_and_evaluate_scratch(subset_name, train_ds, val_ds, test_ds

    print(f"\n--- Training Scratch Model on {subset_name} ---")

    keras.backend.clear_session() # Resets model
    model = build_model_from_scratch()

    checkpoint_filepath = f"{subset_name}_scratch.keras"
    callbacks = [
        keras.callbacks.ModelCheckpoint(
            filepath=checkpoint_filepath,
            save_best_only=True,
            monitor="val_loss")
    ]

    # Training for 100 epochs, as provided in the sample notebook
    history = model.fit(
        train_ds,
        epochs=100,
        validation_data=val_ds,
        callbacks=callbacks,
        verbose=1)

    print("Training complete. Plotting results...")
    plot_history(history, f"Scratch Model: {subset_name}")

    # Loading the best performing model (saved by ModelCheckpoint)
    print("Evaluating best model on test set...")
    test_model = keras.models.load_model(checkpoint_filepath)
    test_loss, test_acc = test_model.evaluate(test_ds)
    print(f"Test accuracy for {subset_name}: {test_acc:.4f}")

    return test_acc
```

```
    return test_acc
```

```
# Step 1: Train=1000, Val=500, Test=500
results["scratch_1000"] = train_and_evaluate_scratch(
    "subset_1000_500_500", train_ds_1, val_ds_1, test_ds_1)

# Step 2: Train=2000, Val=500, Test=500
results["scratch_2000"] = train_and_evaluate_scratch(
    "subset_2000_500_500", train_ds_2, val_ds_2, test_ds_2)

# Step 3: Train=2000, Val=1000, Test=1000 (The "ideal" from sample)
results["scratch_ideal"] = train_and_evaluate_scratch(
    "subset_2000_1000_1000", train_ds_3, val_ds_3, test_ds_3)

print("\nScratch Model Results So Far:")
print(results)
```

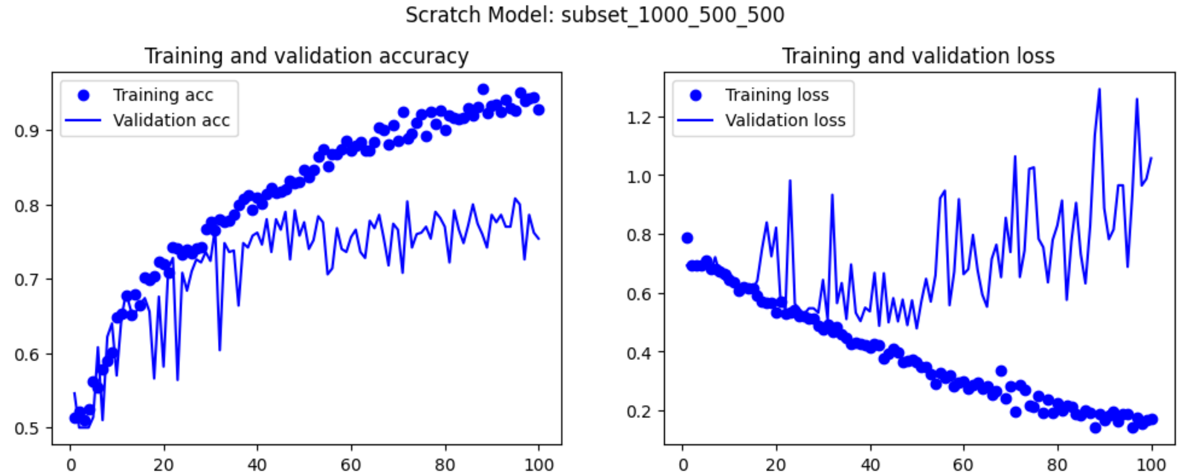
```
--- Training Scratch Model on subset_1000_500_500 ---
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 79ms/step - accuracy: 0.5115 - loss: 0.8
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.4980 - loss: 0.6
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.5215 - loss: 0.6
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step - accuracy: 0.5213 - loss: 0.6
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 62ms/step - accuracy: 0.5507 - loss: 0.7
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.5371 - loss: 0.6
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.5861 - loss: 0.6
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.5711 - loss: 0.6
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 90ms/step - accuracy: 0.6121 - loss: 0.6
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.6585 - loss: 0.6
Epoch 11/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.6350 - loss: 0.6
Epoch 12/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.6802 - loss: 0.6
Epoch 13/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.6504 - loss: 0.6
Epoch 14/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.6909 - loss: 0.6
Epoch 15/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 107ms/step - accuracy: 0.6587 - loss: 0.
Epoch 16/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.6885 - loss: 0.5
Epoch 17/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.7091 - loss: 0.5
Epoch 18/100
```

```
Epoch 18/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.7073 - loss: 0.5
Epoch 19/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.6833 - loss: 0.6
Epoch 20/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.7387 - loss: 0.5
Epoch 21/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.6944 - loss: 0.5
Epoch 22/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.7519 - loss: 0.5
Epoch 23/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.7465 - loss: 0.5
Epoch 24/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.7094 - loss: 0.5
Epoch 25/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.7335 - loss: 0.5
Epoch 26/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 88ms/step - accuracy: 0.7369 - loss: 0.5
Epoch 27/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - accuracy: 0.7364 - loss: 0.5
Epoch 28/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.7356 - loss: 0.5
Epoch 29/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.7709 - loss: 0.4
Epoch 30/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.7633 - loss: 0.4
Epoch 31/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 100ms/step - accuracy: 0.7666 - loss: 0.
Epoch 32/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.8022 - loss: 0.4
Epoch 33/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.7641 - loss: 0.5
Epoch 34/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.7787 - loss: 0.4
Epoch 35/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.7934 - loss: 0.4
Epoch 36/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.8063 - loss: 0.4
Epoch 37/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 72ms/step - accuracy: 0.7918 - loss: 0.4
Epoch 38/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 81ms/step - accuracy: 0.8168 - loss: 0.4
Epoch 39/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.7630 - loss: 0.4
Epoch 40/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8027 - loss: 0.4
Epoch 41/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.7805 - loss: 0.4
Epoch 42/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8321 - loss: 0.3
Epoch 43/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8279 - loss: 0.3
Epoch 44/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.8262 - loss: 0.3
Epoch 45/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 71ms/step - accuracy: 0.8041 - loss: 0.4
```

```
Epoch 46/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8144 - loss: 0.4
Epoch 47/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8359 - loss: 0.3
Epoch 48/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.8318 - loss: 0.3
Epoch 49/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8320 - loss: 0.3
Epoch 50/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.8581 - loss: 0.3
Epoch 51/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step - accuracy: 0.8345 - loss: 0.3
Epoch 52/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8364 - loss: 0.3
Epoch 53/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8709 - loss: 0.3
Epoch 54/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.8630 - loss: 0.3
Epoch 55/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8481 - loss: 0.3
Epoch 56/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8648 - loss: 0.3
Epoch 57/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 80ms/step - accuracy: 0.8669 - loss: 0.3
Epoch 58/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.8843 - loss: 0.2
Epoch 59/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8679 - loss: 0.3
Epoch 60/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8625 - loss: 0.3
Epoch 61/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8669 - loss: 0.3
Epoch 62/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8897 - loss: 0.2
Epoch 63/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.8602 - loss: 0.3
Epoch 64/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 91ms/step - accuracy: 0.8534 - loss: 0.3
Epoch 65/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 65ms/step - accuracy: 0.8804 - loss: 0.2
Epoch 66/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8928 - loss: 0.2
Epoch 67/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9115 - loss: 0.2
Epoch 68/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8508 - loss: 0.4
Epoch 69/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 83ms/step - accuracy: 0.8931 - loss: 0.2
Epoch 70/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.8703 - loss: 0.3
Epoch 71/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9186 - loss: 0.2
Epoch 72/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.8660 - loss: 0.3
Epoch 73/100
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8962 - loss: 0.2
Epoch 74/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.8948 - loss: 0.2
Epoch 75/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 82ms/step - accuracy: 0.9116 - loss: 0.2
Epoch 76/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 81ms/step - accuracy: 0.8707 - loss: 0.2
Epoch 77/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9133 - loss: 0.2
Epoch 78/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9092 - loss: 0.2
Epoch 79/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9112 - loss: 0.2
Epoch 80/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9039 - loss: 0.2
Epoch 81/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9209 - loss: 0.2
Epoch 82/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 90ms/step - accuracy: 0.9122 - loss: 0.2
Epoch 83/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9368 - loss: 0.1
Epoch 84/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9166 - loss: 0.1
Epoch 85/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9242 - loss: 0.2
Epoch 86/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9306 - loss: 0.1
Epoch 87/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.9380 - loss: 0.1
Epoch 88/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.9704 - loss: 0.1
Epoch 89/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9268 - loss: 0.1
Epoch 90/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9357 - loss: 0.1
Epoch 91/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 58ms/step - accuracy: 0.9322 - loss: 0.1
Epoch 92/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.9298 - loss: 0.1
Epoch 93/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 92ms/step - accuracy: 0.9310 - loss: 0.1
Epoch 94/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9360 - loss: 0.1
Epoch 95/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9329 - loss: 0.1
Epoch 96/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9446 - loss: 0.1
Epoch 97/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9493 - loss: 0.1
Epoch 98/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 114ms/step - accuracy: 0.9298 - loss: 0.
Epoch 99/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 60ms/step - accuracy: 0.9393 - loss: 0.1
Epoch 100/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.9123 - loss: 0.2
Training complete. Plotting results
```

```
Training complete. Plotting results...
```

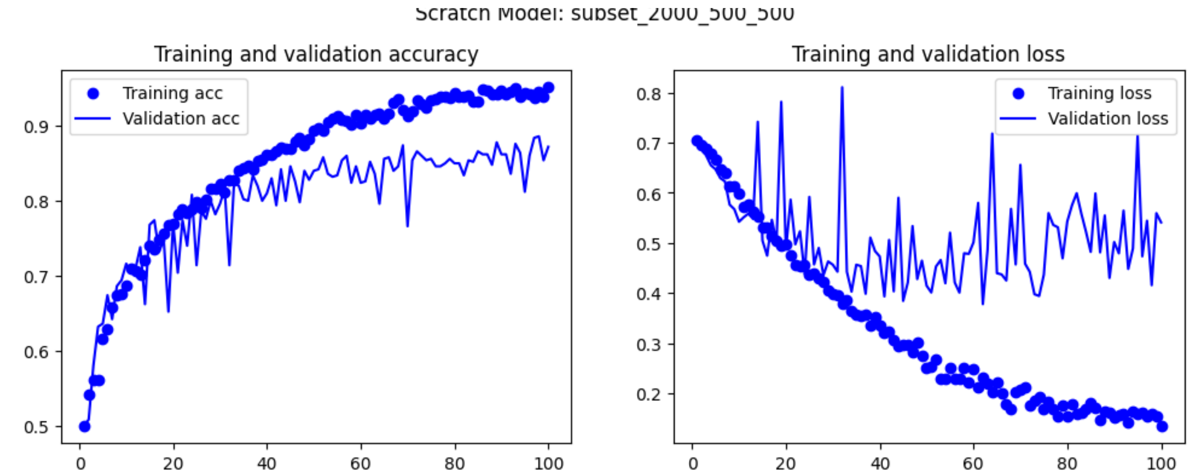



```
Evaluating best model on test set...
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.7479 - loss: 0.6
Test accuracy for subset_1000_500_500: 0.7580

--- Training Scratch Model on subset_2000_500_500 ---
Epoch 1/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 74ms/step - accuracy: 0.4927 - loss: 0.6
Epoch 2/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.5343 - loss: 0.6
Epoch 3/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.5583 - loss: 0.6
Epoch 4/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.5527 - loss: 0.6
Epoch 5/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 67ms/step - accuracy: 0.6105 - loss: 0.6
Epoch 6/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.6306 - loss: 0.6
Epoch 7/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.6552 - loss: 0.6
Epoch 8/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.6804 - loss: 0.6
Epoch 9/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 66ms/step - accuracy: 0.6779 - loss: 0.6
Epoch 10/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.6919 - loss: 0.5
Epoch 11/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.7199 - loss: 0.5
Epoch 12/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.7084 - loss: 0.5
Epoch 13/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 63ms/step - accuracy: 0.6940 - loss: 0.5
Epoch 14/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.7308 - loss: 0.5
Epoch 15/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7456 - loss: 0.5
Epoch 16/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7439 - loss: 0.5
Epoch 17/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.7464 - loss: 0.5
Epoch 18/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.7509 - loss: 0.5
```

```
Epoch 19/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7679 - loss: 0.4
Epoch 20/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7541 - loss: 0.5
Epoch 21/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.7939 - loss: 0.4
Epoch 22/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.7895 - loss: 0.4
Epoch 23/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.7843 - loss: 0.4
Epoch 24/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 67ms/step - accuracy: 0.7860 - loss: 0.4
Epoch 25/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.8085 - loss: 0.4
Epoch 26/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 48ms/step - accuracy: 0.7720 - loss: 0.4
Epoch 27/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 70ms/step - accuracy: 0.7999 - loss: 0.4
Epoch 28/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.8190 - loss: 0.4
Epoch 29/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.8184 - loss: 0.3
Epoch 30/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 63ms/step - accuracy: 0.8271 - loss: 0.4
Epoch 31/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 49ms/step - accuracy: 0.8255 - loss: 0.3
Epoch 32/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.8339 - loss: 0.3
Epoch 33/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 60ms/step - accuracy: 0.8272 - loss: 0.3
Epoch 34/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 49ms/step - accuracy: 0.8380 - loss: 0.3
Epoch 35/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 51ms/step - accuracy: 0.8390 - loss: 0.3
Epoch 36/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.8398 - loss: 0.3
Epoch 37/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8486 - loss: 0.3
Epoch 38/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.8656 - loss: 0.3
Epoch 39/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.8593 - loss: 0.3
Epoch 40/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 70ms/step - accuracy: 0.8615 - loss: 0.3
Epoch 41/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.8621 - loss: 0.3
Epoch 42/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.8555 - loss: 0.3
Epoch 43/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 76ms/step - accuracy: 0.8710 - loss: 0.3
Epoch 44/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.8659 - loss: 0.3
Epoch 45/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 48ms/step - accuracy: 0.8662 - loss: 0.2
Epoch 46/100
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 62ms/step - accuracy: 0.8825 - loss: 0.2
Epoch 47/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.8842 - loss: 0.2
Epoch 48/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.8763 - loss: 0.2
Epoch 49/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.8767 - loss: 0.2
Epoch 50/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 70ms/step - accuracy: 0.8931 - loss: 0.2
Epoch 51/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 49ms/step - accuracy: 0.8959 - loss: 0.2
Epoch 52/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.8860 - loss: 0.2
Epoch 53/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 71ms/step - accuracy: 0.8994 - loss: 0.2
Epoch 54/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.9156 - loss: 0.2
Epoch 55/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 47ms/step - accuracy: 0.9197 - loss: 0.2
Epoch 56/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 67ms/step - accuracy: 0.9166 - loss: 0.2
Epoch 57/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.9004 - loss: 0.2
Epoch 58/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9146 - loss: 0.2
Epoch 59/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9129 - loss: 0.2
Epoch 60/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.9112 - loss: 0.2
Epoch 61/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.9136 - loss: 0.2
Epoch 62/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.9142 - loss: 0.2
Epoch 63/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.9189 - loss: 0.2
Epoch 64/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.9160 - loss: 0.2
Epoch 65/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 48ms/step - accuracy: 0.9084 - loss: 0.2
Epoch 66/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9094 - loss: 0.2
Epoch 67/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.9370 - loss: 0.1
Epoch 68/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 48ms/step - accuracy: 0.9387 - loss: 0.1
Epoch 69/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.9123 - loss: 0.2
Epoch 70/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9130 - loss: 0.2
Epoch 71/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.9214 - loss: 0.1
Epoch 72/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9379 - loss: 0.1
Epoch 73/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 51ms/step - accuracy: 0.9351 - loss: 0.1
```

```
Epoch 74/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9269 - loss: 0.1
Epoch 75/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9328 - loss: 0.1
Epoch 76/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 67ms/step - accuracy: 0.9404 - loss: 0.1
Epoch 77/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.9405 - loss: 0.1
Epoch 78/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9433 - loss: 0.1
Epoch 79/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 63ms/step - accuracy: 0.9369 - loss: 0.1
Epoch 80/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9465 - loss: 0.1
Epoch 81/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9366 - loss: 0.1
Epoch 82/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9483 - loss: 0.1
Epoch 83/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.9382 - loss: 0.1
Epoch 84/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.9307 - loss: 0.1
Epoch 85/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9319 - loss: 0.1
Epoch 86/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9554 - loss: 0.1
Epoch 87/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 67ms/step - accuracy: 0.9460 - loss: 0.1
Epoch 88/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.9421 - loss: 0.1
Epoch 89/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.9472 - loss: 0.1
Epoch 90/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.9482 - loss: 0.1
Epoch 91/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.9510 - loss: 0.1
Epoch 92/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9528 - loss: 0.1
Epoch 93/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 62ms/step - accuracy: 0.9518 - loss: 0.1
Epoch 94/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.9398 - loss: 0.1
Epoch 95/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.9422 - loss: 0.1
Epoch 96/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9533 - loss: 0.1
Epoch 97/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 66ms/step - accuracy: 0.9402 - loss: 0.1
Epoch 98/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.9496 - loss: 0.1
Epoch 99/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.9410 - loss: 0.1
Epoch 100/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.9541 - loss: 0.1
Training complete. Plotting results...
```



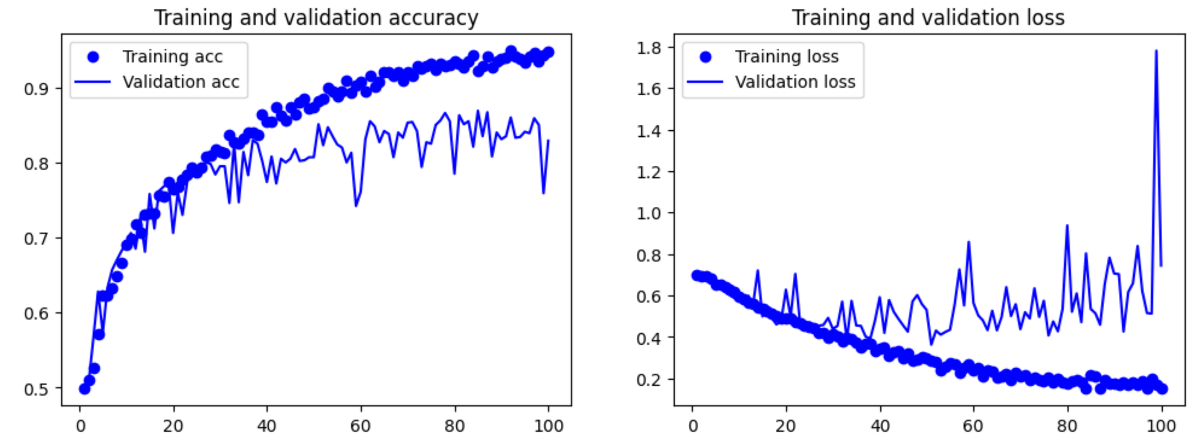

```
Evaluating best model on test set...
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8194 - loss: 0.7
Test accuracy for subset_2000_500_500: 0.8400

--- Training Scratch Model on subset_2000_1000_1000 ---
Epoch 1/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 68ms/step - accuracy: 0.5141 - loss: 0.6
Epoch 2/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.5070 - loss: 0.6
Epoch 3/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 83ms/step - accuracy: 0.5181 - loss: 0.6
Epoch 4/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.5469 - loss: 0.6
Epoch 5/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.6244 - loss: 0.6
Epoch 6/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 77ms/step - accuracy: 0.6174 - loss: 0.6
Epoch 7/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.6304 - loss: 0.6
Epoch 8/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.6580 - loss: 0.6
Epoch 9/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 82ms/step - accuracy: 0.6727 - loss: 0.6
Epoch 10/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 55ms/step - accuracy: 0.6848 - loss: 0.5
Epoch 11/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 75ms/step - accuracy: 0.6862 - loss: 0.5
Epoch 12/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 58ms/step - accuracy: 0.7017 - loss: 0.5
Epoch 13/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.6889 - loss: 0.5
Epoch 14/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.7367 - loss: 0.5
Epoch 15/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.7208 - loss: 0.5
Epoch 16/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 55ms/step - accuracy: 0.7299 - loss: 0.5
Epoch 17/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 83ms/step - accuracy: 0.7513 - loss: 0.5
Epoch 18/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.7564 - loss: 0.5
Epoch 19/100
```

```
Epoch 19/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.7811 - loss: 0.4
Epoch 20/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 77ms/step - accuracy: 0.7604 - loss: 0.5
Epoch 21/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.7563 - loss: 0.4
Epoch 22/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.7760 - loss: 0.4
Epoch 23/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.7721 - loss: 0.4
Epoch 24/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - accuracy: 0.7860 - loss: 0.4
Epoch 25/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.7779 - loss: 0.4
Epoch 26/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 80ms/step - accuracy: 0.7973 - loss: 0.4
Epoch 27/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.7997 - loss: 0.4
Epoch 28/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - accuracy: 0.8057 - loss: 0.4
Epoch 29/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 79ms/step - accuracy: 0.8155 - loss: 0.3
Epoch 30/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8128 - loss: 0.3
Epoch 31/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.8147 - loss: 0.4
Epoch 32/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 99ms/step - accuracy: 0.8298 - loss: 0.3
Epoch 33/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8314 - loss: 0.3
Epoch 34/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8295 - loss: 0.3
Epoch 35/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 92ms/step - accuracy: 0.8294 - loss: 0.3
Epoch 36/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.8432 - loss: 0.3
Epoch 37/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 55ms/step - accuracy: 0.8474 - loss: 0.3
Epoch 38/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 81ms/step - accuracy: 0.8560 - loss: 0.3
Epoch 39/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.8678 - loss: 0.3
Epoch 40/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8540 - loss: 0.3
Epoch 41/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 80ms/step - accuracy: 0.8515 - loss: 0.3
Epoch 42/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.8759 - loss: 0.3
Epoch 43/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8577 - loss: 0.3
Epoch 44/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 72ms/step - accuracy: 0.8636 - loss: 0.3
Epoch 45/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8777 - loss: 0.2
Epoch 46/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 55ms/step - accuracy: 0.8748 - loss: 0.3
```

```
Epoch 47/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8761 - loss: 0.2
Epoch 48/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.8867 - loss: 0.2
Epoch 49/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 58ms/step - accuracy: 0.8546 - loss: 0.3
Epoch 50/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 81ms/step - accuracy: 0.8724 - loss: 0.3
Epoch 51/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.8817 - loss: 0.2
Epoch 52/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - accuracy: 0.8852 - loss: 0.2
Epoch 53/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 90ms/step - accuracy: 0.9140 - loss: 0.2
Epoch 54/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8982 - loss: 0.2
Epoch 55/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8925 - loss: 0.2
Epoch 56/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 100ms/step - accuracy: 0.8844 - loss: 0.
Epoch 57/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9081 - loss: 0.2
Epoch 58/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.8903 - loss: 0.2
Epoch 59/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 75ms/step - accuracy: 0.8942 - loss: 0.2
Epoch 60/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9119 - loss: 0.2
Epoch 61/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.8940 - loss: 0.2
Epoch 62/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 81ms/step - accuracy: 0.9188 - loss: 0.2
Epoch 63/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9013 - loss: 0.2
Epoch 64/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 55ms/step - accuracy: 0.9089 - loss: 0.2
Epoch 65/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 80ms/step - accuracy: 0.9241 - loss: 0.1
Epoch 66/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9246 - loss: 0.1
Epoch 67/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.9222 - loss: 0.2
Epoch 68/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9126 - loss: 0.2
Epoch 69/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9022 - loss: 0.2
Epoch 70/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.9162 - loss: 0.2
Epoch 71/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 81ms/step - accuracy: 0.9092 - loss: 0.2
Epoch 72/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9338 - loss: 0.1
Epoch 73/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.9259 - loss: 0.1
Epoch 74/100
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 76ms/step - accuracy: 0.9284 - loss: 0.1
Epoch 75/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.9273 - loss: 0.1
Epoch 76/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9327 - loss: 0.1
Epoch 77/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.9338 - loss: 0.1
Epoch 78/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 81ms/step - accuracy: 0.9345 - loss: 0.1
Epoch 79/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9320 - loss: 0.1
Epoch 80/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - accuracy: 0.9331 - loss: 0.1
Epoch 81/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 92ms/step - accuracy: 0.9344 - loss: 0.1
Epoch 82/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.9263 - loss: 0.1
Epoch 83/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9361 - loss: 0.1
Epoch 84/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 85ms/step - accuracy: 0.9529 - loss: 0.1
Epoch 85/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - accuracy: 0.9123 - loss: 0.2
Epoch 86/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.9341 - loss: 0.1
Epoch 87/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 82ms/step - accuracy: 0.9368 - loss: 0.1
Epoch 88/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.9282 - loss: 0.1
Epoch 89/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9340 - loss: 0.1
Epoch 90/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9283 - loss: 0.2
Epoch 91/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.9459 - loss: 0.1
Epoch 92/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.9502 - loss: 0.1
Epoch 93/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 70ms/step - accuracy: 0.9393 - loss: 0.1
Epoch 94/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.9376 - loss: 0.1
Epoch 95/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9276 - loss: 0.1
Epoch 96/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.9439 - loss: 0.1
Epoch 97/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 77ms/step - accuracy: 0.9470 - loss: 0.1
Epoch 98/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.9372 - loss: 0.1
Epoch 99/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 64ms/step - accuracy: 0.9450 - loss: 0.1
Epoch 100/100
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 77ms/step - accuracy: 0.9413 - loss: 0.1
Training complete. Plotting results...
```

Scratch Model: subset_2000_1000_1000




```
Evaluating best model on test set...
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.8223 - loss: 0.4
Test accuracy for subset_2000_1000_1000: 0.8250

Scratch Model Results So Far:
{'scratch_1000': 0.7580000162124634, 'scratch_2000': 0.839999973773956
```

From Step 1 to 2, Increasing training data from 1,000 → 2,000 improved test accuracy substantially (0.758 → 0.840). In Step 3, The 'ideal' run used the same training amount as Step 2 (2,000) but larger validation/test splits; test accuracy for that run was slightly lower (0.825), likely because the larger test set is a stricter estimate and/or because of variance in which images were selected.

```
def build_and_train_pretrained(subset_name, train_ds, val_ds, test_ds

    print(f"\n--- Training Pretrained Model on {subset_name} ---")

    keras.backend.clear_session()

    # 1. Loading VGG16 base, frozen
    conv_base = keras.applications.vgg16.VGG16(
        weights="imagenet",
        include_top=False,
        input_shape=(180, 180, 3))
    conv_base.trainable = False

    # 2. Adding augmentation, classifier head
    data_augmentation = keras.Sequential(
        [layers.RandomFlip("horizontal"),
         layers.RandomRotation(0.1),
         layers.RandomZoom(0.2)]
    )

    inputs = keras.Input(shape=(180, 180, 3))
    x = data_augmentation(inputs)
    x = keras.applications.vgg16.preprocess_input(x)
    x = conv_base(x)
    x = layers.Flatten()(x)
    x = layers.Dense(256)(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)
    model = keras.Model(inputs, outputs)

    # 3. --- PHASE 1: FEATURE EXTRACTION ---
    print("... Phase 1: Training classifier head ...")
    checkpoint_filepath_phase1 = f"{subset_name}_pretrained_phase1.ke
    callbacks_phase1 = [
        keras.callbacks.ModelCheckpoint(
            filepath=checkpoint_filepath_phase1,
            save_best_only=True,
            monitor="val_loss")
    ]

    model.compile(loss="binary_crossentropy",
                  optimizer="rmsprop",
                  metrics=["accuracy"])

    history_phase1 = model.fit(
        train_ds,
        epochs=50, # Training classifier for 50 epochs
        validation_data=val_ds,
        callbacks=callbacks_phase1
```

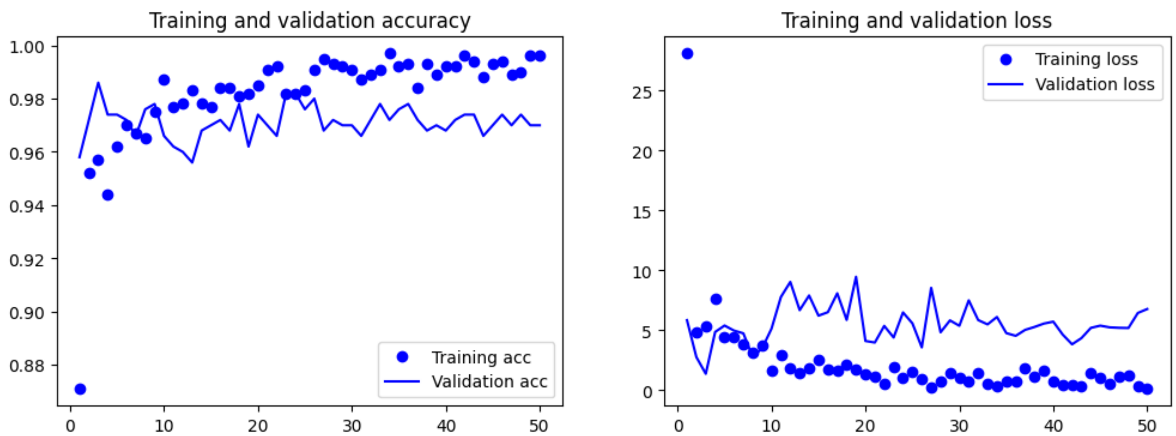
```
        callbacks=callbacks_phase1,
        verbose=0)

    print("Phase 1 complete. Plotting results...")
    plot_history(history_phase1, f"Pretrained (Phase 1): {subset_name

    # 4. --- PHASE 2: FINE-TUNING ---
    print("... Phase 2: Fine-tuning ...")
    model = keras.models.load_model(checkpoint_filepath_phase1) # Loa

    # Unfreeze the top layers of VGG16
    conv_base.trainable = True
    for layer in conv_base.layers[:-4]:
        layer.trainable = False

    checkpoint_filepath_phase2 = f"{subset_name}_pretrained_phase2.ke
    callbacks_phase2 = [
        keras.callbacks.ModelCheckpoint(
            filepath=checkpoint_filepath_phase2,
            save_best_only=True,
            monitor="val_loss")
    ]

    # Re-compiling with a very low learning rate
    model.compile(loss="binary_crossentropy",
                  optimizer=keras.optimizers.RMSprop(learning_rate=1e
                  metrics=["accuracy"])

    history_phase2 = model.fit(
        train_ds,
        epochs=30, # Fine-tune for 30 epochs
        validation_data=val_ds,
        callbacks=callbacks_phase2,
        verbose=0)

    print("Phase 2 complete. Plotting results...")
    plot_history(history_phase2, f"Pretrained (Phase 2 Fine-Tune): {s

    # 5. --- FINAL EVALUATION ---
    print("Evaluating best model on test set...")
    test_model = keras.models.load_model(checkpoint_filepath_phase2)
    test_loss, test_acc = test_model.evaluate(test_ds)
    print(f"Test accuracy for {subset_name}: {test_acc:.4f}")

    return test_acc
```

```
# --- Running the Pretrained Experiments ---

# Step 4 (part 1): Train=1000, Val=500, Test=500
results["pretrained_1000"] = build_and_train_pretrained(
    "subset_1000_500_500", train_ds_1, val_ds_1, test_ds_1)
```

```
    "subset_1000_500_500", train_ds_1, val_ds_1, test_ds_1)

# Step 4 (part 2): Train=2000, Val=500, Test=500
results["pretrained_2000"] = build_and_train_pretrained(
    "subset_2000_500_500", train_ds_2, val_ds_2, test_ds_2)

# Step 4 (part 3): Train=2000, Val=1000, Test=1000 (The "ideal")
results["pretrained_ideal"] = build_and_train_pretrained(
    "subset_2000_1000_1000", train_ds_3, val_ds_3, test_ds_3)
```

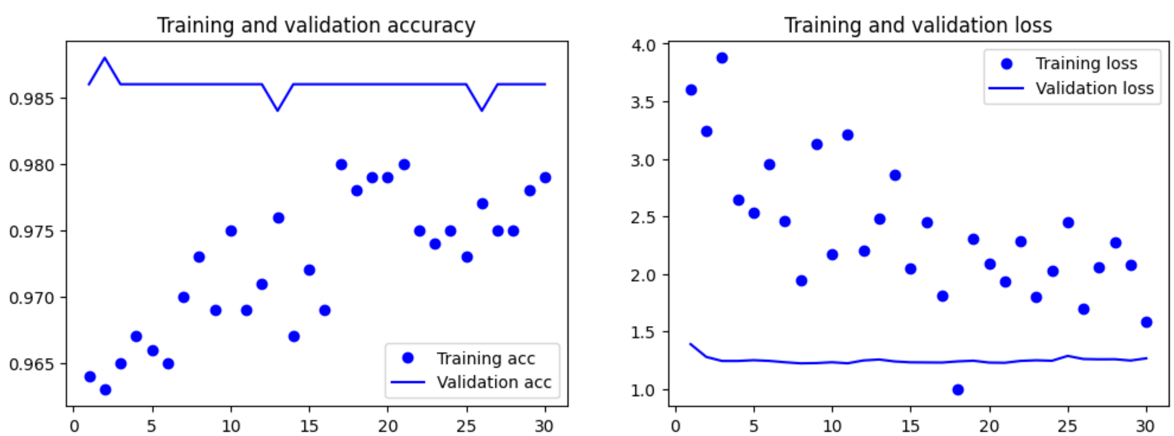
```
--- Training Pretrained Model on subset_1000_500_500 ---
Downloading data from https://storage.googleapis.com/tensorflow/keras-a
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
... Phase 1: Training classifier head ...
Phase 1 complete. Plotting results...
```

Pretrained (Phase 1): subset_1000_500_500




```
... Phase 2: Fine-tuning ...
Phase 2 complete. Plotting results...
```

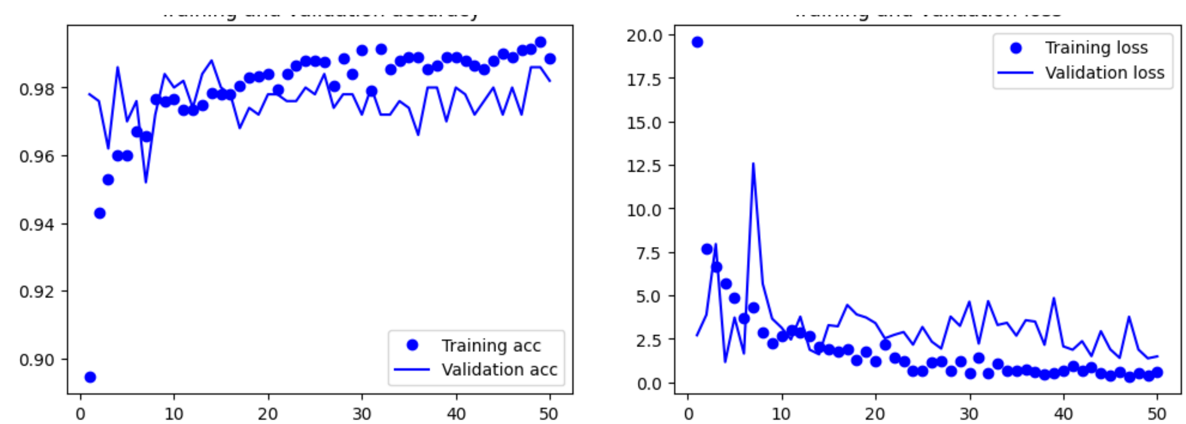
Pretrained (Phase 2 Fine-Tune): subset_1000_500_500




```
Evaluating best model on test set...
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 95ms/step - accuracy: 0.9635 - loss: 7.9
Test accuracy for subset_1000_500_500: 0.9660

--- Training Pretrained Model on subset_2000_500_500 ---
... Phase 1: Training classifier head ...
Phase 1 complete. Plotting results...
```

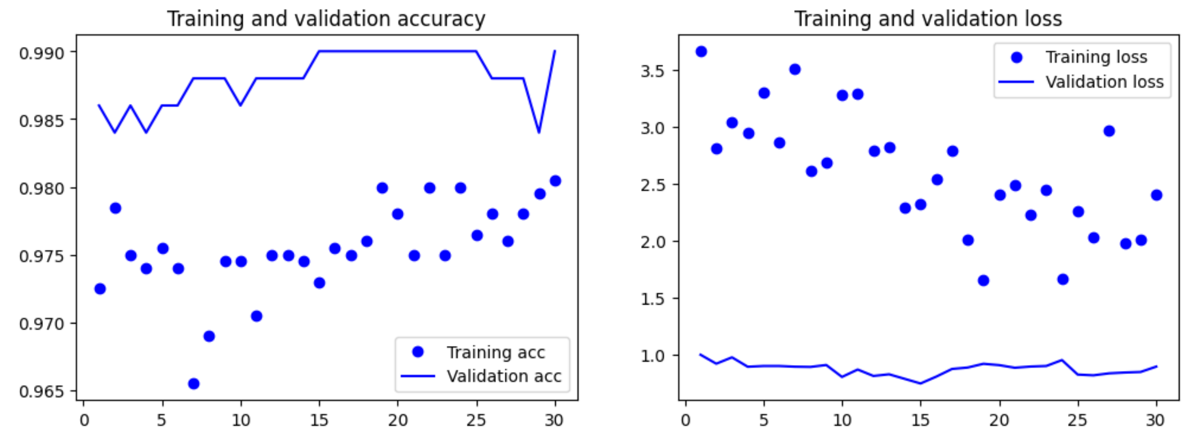
Pretrained (Phase 1): subset_2000_500_500

Training and validation accuracy Training and validation loss



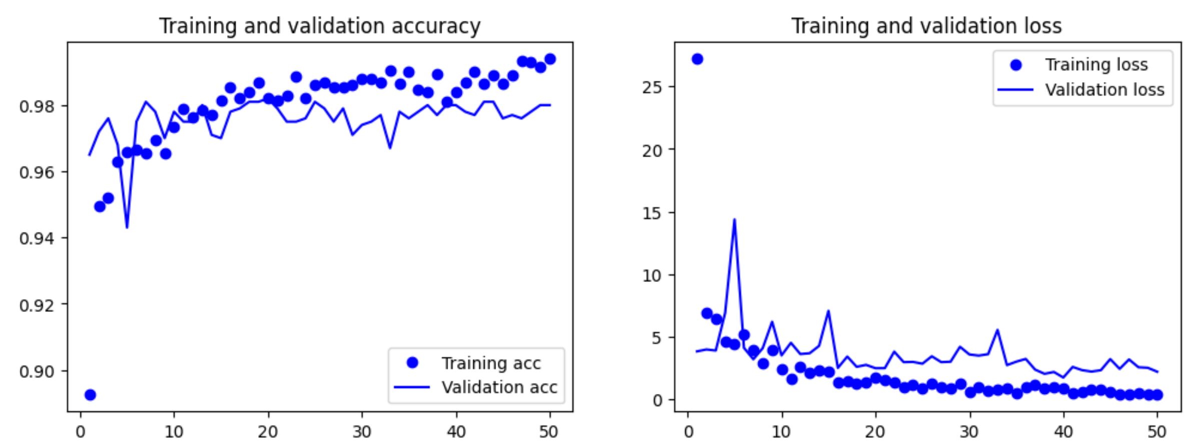

```
... Phase 2: Fine-tuning ...
Phase 2 complete. Plotting results...
```

Pretrained (Phase 2 Fine-Tune): subset_2000_500_500



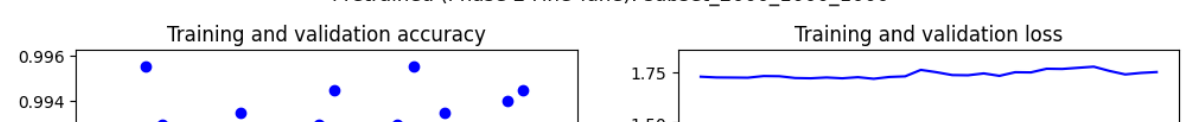

```
Evaluating best model on test set...
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 96ms/step - accuracy: 0.9690 - loss: 4.3
Test accuracy for subset_2000_500_500: 0.9740

--- Training Pretrained Model on subset_2000_1000_1000 ---
... Phase 1: Training classifier head ...
Phase 1 complete. Plotting results...
```

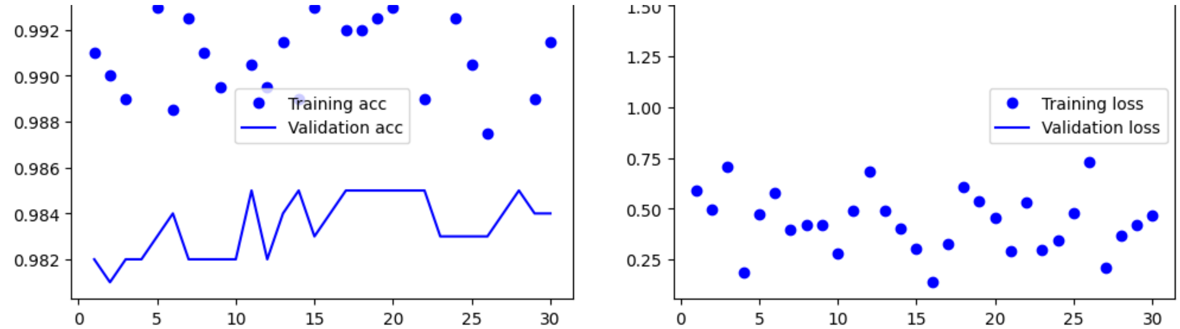
Pretrained (Phase 1): subset_2000_1000_1000




```
... Phase 2: Fine-tuning ...
Phase 2 complete. Plotting results...
```

Pretrained (Phase 2 Fine-Tune): subset_2000_1000_1000






```
Evaluating best model on test set...
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step - accuracy: 0.9870 - loss: 1.45
Test accuracy for subset_2000_1000_1000: 0.9830
```

In Step 4, Transfer learning (pretrained VGG16 + fine-tuning) outperformed the model trained from scratch by a large margin for all sample sizes. Even with only 1,000 training images, the pretrained model reached ~96.6% test accuracy, whereas the scratch model reached ~75.8%. Also noticeably, Pretrained models improved significantly with more train data (96.6% → 97.4% → 98.3%), indicating diminishing returns but still measurable gains from additional labeled data when combined with transfer learning and fine-tuning.

```
print("\n\n--- FINAL RESULTS ---")
print(results)
```

```
--- FINAL RESULTS ---
{'scratch_1000': 0.7580000162124634, 'scratch_2000': 0.839999973773956
```