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

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

```
!unzip /content/drive/MyDrive/cats_vs_dogs_small.zip
```

```
  inflating: cats_vs_dogs_small/validation/dogs/1442.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1443.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1444.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1445.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1446.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1447.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1448.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1449.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1450.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1451.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1452.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1453.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1454.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1455.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1456.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1457.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1458.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1459.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1460.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1461.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1462.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1463.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1464.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1465.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1466.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1467.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1468.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1469.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1470.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1471.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1472.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1473.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1474.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1475.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1476.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1477.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1478.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1479.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1480.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1481.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1482.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1483.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1484.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1485.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1486.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1487.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1488.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1489.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1490.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1491.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1492.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1493.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1494.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1495.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1496.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1497.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1498.jpg
  inflating: cats_vs_dogs_small/validation/dogs/1499.jpg
```

```
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.utils import image_dataset_from_directory

# === Define directory and load datasets ===
data_dir = "/content/cats_vs_dogs_small"



# Full training dataset
complete_train_data = image_dataset_from_directory(
    os.path.join(data_dir, "train")
```

```
    os.path.join(data_dir, "train"),
    image_size=(160, 160),  # Slightly different image size
    batch_size=32,
    shuffle=True,
    seed=123  # Different seed
)
```

```
Found 2000 files belonging to 2 classes.
```

```
# Validation and test datasets
val_data = image_dataset_from_directory(
    os.path.join(data_dir, "validation"),
    image_size=(160, 160),
    batch_size=32
)

test_data = image_dataset_from_directory(
    os.path.join(data_dir, "test"),
    image_size=(160, 160),
    batch_size=32
)
```

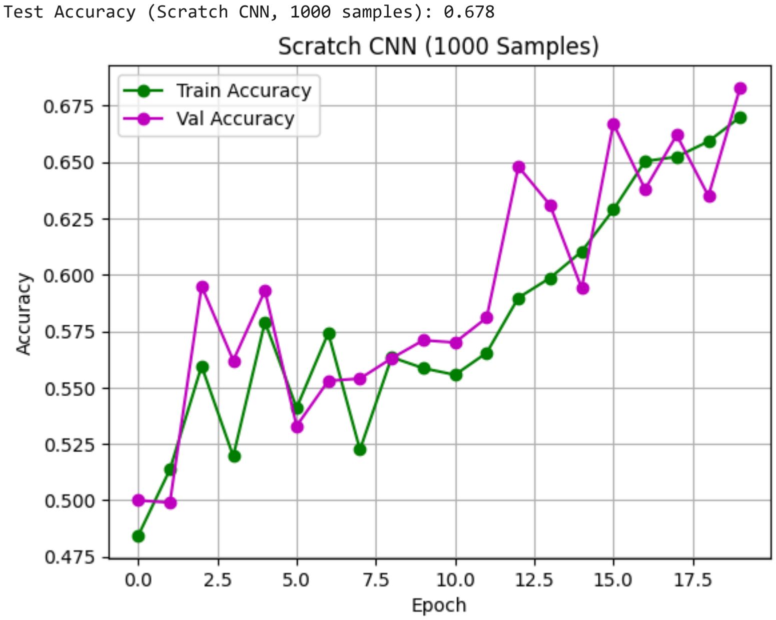
```
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

=== Step 1: CNN from Scratch with 1000 Training Samples ===

```
# === Step 1: CNN from Scratch with 1000 Training Samples ===
train_data_1000 = complete_train_data.take(32)  # Approx. 1000 samples (32 batches)

# Data augmentation layer
augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.15),  # Slightly different rotation factor
    layers.RandomZoom(0.25)       # Slightly different zoom
])

# Define CNN architecture
input_layer = keras.Input(shape=(160, 160, 3))
x = augmentation(input_layer)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(32, 3, activation="relu", padding="same")(x)  # Added padding
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(64, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(128, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(256, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.4)(x)  # Slightly different dropout rate
output_layer = layers.Dense(1, activation="sigmoid")(x)

cnn_model = keras.Model(input_layer, output_layer)
cnn_model.compile(optimizer="adam",  # Different optimizer
                  loss="binary_crossentropy",
                  metrics=["accuracy"])

# Train the model
training_history = cnn_model.fit(train_data_1000, epochs=20, validation_data=val_data)

# Evaluate on test set
test_loss_scratch_1000, test_acc_scratch_1000 = cnn_model.evaluate(test_data)
print(f"Test Accuracy (Scratch CNN, 1000 samples): {test_acc_scratch_1000:.3f}")

# Plot results
train_acc = training_history.history["accuracy"]
val_acc = training_history.history["val_accuracy"]
plt.figure()
plt.plot(train_acc, "go-", label="Train Accuracy")
plt.plot(val_acc, "mo-", label="Val Accuracy")
plt.title("Scratch CNN (1000 Samples)")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
```

```
plt.legend()
plt.grid(True)
plt.savefig("scratch_cnn_1000.png")
plt.show()
```

```
Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 127ms/step - accuracy: 0.4881 - loss: 0.7125 - val_accuracy: 0.5000 - val_loss: 0.6921
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.4963 - loss: 0.6935 - val_accuracy: 0.4990 - val_loss: 0.6906
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.5371 - loss: 0.6914 - val_accuracy: 0.5950 - val_loss: 0.6768
Epoch 4/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.5474 - loss: 0.6861 - val_accuracy: 0.5620 - val_loss: 0.6830
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.5530 - loss: 0.6796 - val_accuracy: 0.5930 - val_loss: 0.6607
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.5615 - loss: 0.6854 - val_accuracy: 0.5330 - val_loss: 0.6766
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 96ms/step - accuracy: 0.5451 - loss: 0.6758 - val_accuracy: 0.5530 - val_loss: 0.7323
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.5313 - loss: 0.7081 - val_accuracy: 0.5540 - val_loss: 0.6859
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.5522 - loss: 0.6849 - val_accuracy: 0.5630 - val_loss: 0.6793
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.5719 - loss: 0.6870 - val_accuracy: 0.5710 - val_loss: 0.6780
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.5506 - loss: 0.6832 - val_accuracy: 0.5700 - val_loss: 0.6753
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 118ms/step - accuracy: 0.5711 - loss: 0.6764 - val_accuracy: 0.5810 - val_loss: 0.6676
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.5871 - loss: 0.6728 - val_accuracy: 0.6480 - val_loss: 0.6562
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.5832 - loss: 0.6644 - val_accuracy: 0.6310 - val_loss: 0.6394
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.6036 - loss: 0.6613 - val_accuracy: 0.5940 - val_loss: 0.6491
Epoch 16/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.6118 - loss: 0.6580 - val_accuracy: 0.6670 - val_loss: 0.6171
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 98ms/step - accuracy: 0.6324 - loss: 0.6359 - val_accuracy: 0.6380 - val_loss: 0.6219
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 85ms/step - accuracy: 0.6550 - loss: 0.6290 - val_accuracy: 0.6620 - val_loss: 0.6236
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.6675 - loss: 0.6238 - val_accuracy: 0.6350 - val_loss: 0.6287
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.6483 - loss: 0.6336 - val_accuracy: 0.6830 - val_loss: 0.6003
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.6843 - loss: 0.6097
Test Accuracy (Scratch CNN, 1000 samples): 0.678
```

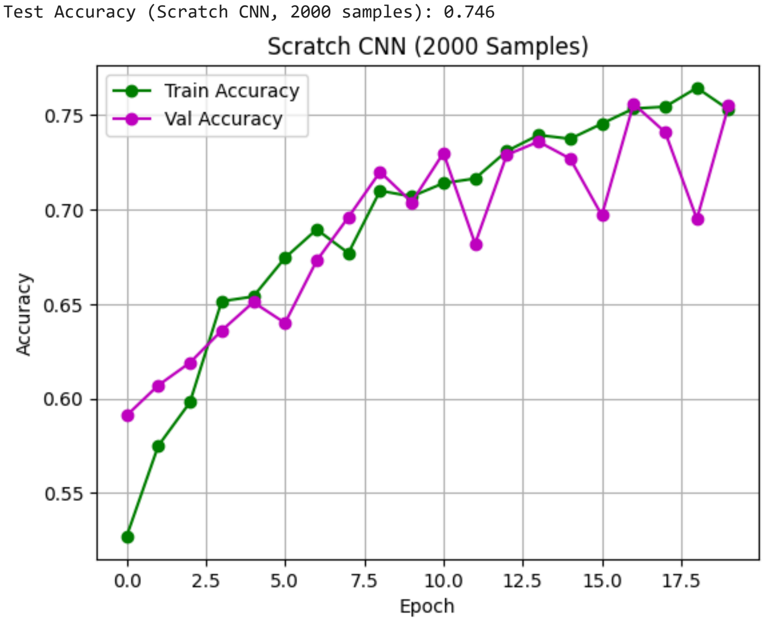


=== Step 2: CNN from Scratch with 2000 Training Samples ===

```
# === Step 2: CNN from Scratch with 2000 Training Samples ===
train_data_2000 = complete_train_data.take(63)  # Approx. 2000 samples (63 batches)
```

```
# Reuse augmentation and redefine model
input_layer = keras.Input(shape=(160, 160, 3))
x = augmentation(input_layer)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(32, 3, activation="relu", padding="same")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(64, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(128, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(256, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.4)(x)
output_layer = layers.Dense(1, activation="sigmoid")(x)

cnn_model_2000 = keras.Model(input_layer, output_layer)
cnn_model_2000.compile(optimizer="adam",
                       loss="binary_crossentropy",
                       metrics=["accuracy"])

# Train and evaluate
training_history_2000 = cnn_model_2000.fit(train_data_2000, epochs=20, validation_data=val_data)
test_loss_scratch_2000, test_acc_scratch_2000 = cnn_model_2000.evaluate(test_data)
print(f"Test Accuracy (Scratch CNN, 2000 samples): {test_acc_scratch_2000:.3f}")

# Plot
train_acc_2000 = training_history_2000.history["accuracy"]
val_acc_2000 = training_history_2000.history["val_accuracy"]
plt.figure()
plt.plot(train_acc_2000, "go-", label="Train Accuracy")
plt.plot(val_acc_2000, "mo-", label="Val Accuracy")
plt.title("Scratch CNN (2000 Samples)")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.savefig("scratch_cnn_2000.png")
plt.show()
```

```
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.4919 - loss: 0.6961 - val_accuracy: 0.5910 - val_loss: 0.6753
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.5672 - loss: 0.6839 - val_accuracy: 0.6070 - val_loss: 0.6666
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.5796 - loss: 0.6728 - val_accuracy: 0.6190 - val_loss: 0.6433
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.6281 - loss: 0.6420 - val_accuracy: 0.6360 - val_loss: 0.6596
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 76ms/step - accuracy: 0.6578 - loss: 0.6323 - val_accuracy: 0.6510 - val_loss: 0.6186
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.6670 - loss: 0.6206 - val_accuracy: 0.6400 - val_loss: 0.6555
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.6892 - loss: 0.6018 - val_accuracy: 0.6730 - val_loss: 0.5968
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.6436 - loss: 0.6119 - val_accuracy: 0.6960 - val_loss: 0.5591
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 51ms/step - accuracy: 0.6964 - loss: 0.5818 - val_accuracy: 0.7200 - val_loss: 0.5624
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 47ms/step - accuracy: 0.7047 - loss: 0.5840 - val_accuracy: 0.7040 - val_loss: 0.5608
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 80ms/step - accuracy: 0.6949 - loss: 0.5858 - val_accuracy: 0.7300 - val_loss: 0.5445
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.7041 - loss: 0.5626 - val_accuracy: 0.6820 - val_loss: 0.6662
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7019 - loss: 0.5652 - val_accuracy: 0.7290 - val_loss: 0.5335
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - accuracy: 0.7246 - loss: 0.5510 - val_accuracy: 0.7360 - val_loss: 0.5111
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.7405 - loss: 0.5288 - val_accuracy: 0.7270 - val_loss: 0.5332
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.7180 - loss: 0.5583 - val_accuracy: 0.6970 - val_loss: 0.5883
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.7426 - loss: 0.5290 - val_accuracy: 0.7560 - val_loss: 0.5045
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.7327 - loss: 0.5417 - val_accuracy: 0.7410 - val_loss: 0.5256
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 78ms/step - accuracy: 0.7637 - loss: 0.4936 - val_accuracy: 0.6950 - val_loss: 0.6088
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.7276 - loss: 0.5337 - val_accuracy: 0.7550 - val_loss: 0.5020
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.7302 - loss: 0.5848
Test Accuracy (Scratch CNN, 2000 samples): 0.746
```

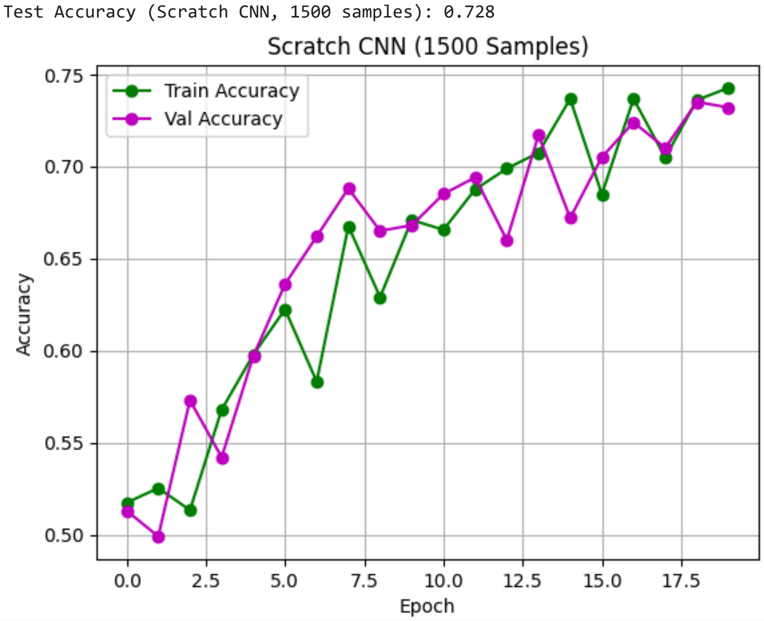


=== Step 3: CNN from Scratch with 1500 Training Samples ===

```
# === Step 3: CNN from Scratch with 1500 Training Samples ===
train_data_1500 = complete_train_data.take(47)  # Approx. 1500 samples (47 batches)

# Define model again
input_layer = keras.Input(shape=(160, 160, 3))
x = augmentation(input_layer)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(32, 3, activation="relu", padding="same")(x)
```

```
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(64, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(128, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(256, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.4)(x)
output_layer = layers.Dense(1, activation="sigmoid")(x)

cnn_model_1500 = keras.Model(input_layer, output_layer)
cnn_model_1500.compile(optimizer="adam",
                       loss="binary_crossentropy",
                       metrics=["accuracy"])

# Train and evaluate
training_history_1500 = cnn_model_1500.fit(train_data_1500, epochs=20, validation_data=val_data)
test_loss_scratch_1500, test_acc_scratch_1500 = cnn_model_1500.evaluate(test_data)
print(f"Test Accuracy (Scratch CNN, 1500 samples): {test_acc_scratch_1500:.3f}")

# Plot
train_acc_1500 = training_history_1500.history["accuracy"]
val_acc_1500 = training_history_1500.history["val_accuracy"]
plt.figure()
plt.plot(train_acc_1500, "go-", label="Train Accuracy")
plt.plot(val_acc_1500, "mo-", label="Val Accuracy")
plt.title("Scratch CNN (1500 Samples)")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.savefig("scratch_cnn_1500.png")
plt.show()
```

```
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 59ms/step - accuracy: 0.4898 - loss: 0.7005 - val_accuracy: 0.5130 - val_loss: 0.6916
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.5380 - loss: 0.6922 - val_accuracy: 0.4990 - val_loss: 0.6923
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 86ms/step - accuracy: 0.5093 - loss: 0.6911 - val_accuracy: 0.5730 - val_loss: 0.6762
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.5624 - loss: 0.6816 - val_accuracy: 0.5420 - val_loss: 0.7143
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.5789 - loss: 0.6722 - val_accuracy: 0.5970 - val_loss: 0.6524
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.6206 - loss: 0.6549 - val_accuracy: 0.6360 - val_loss: 0.6367
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 98ms/step - accuracy: 0.5701 - loss: 0.6913 - val_accuracy: 0.6620 - val_loss: 0.6411
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.6566 - loss: 0.6420 - val_accuracy: 0.6880 - val_loss: 0.5949
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.6277 - loss: 0.6426 - val_accuracy: 0.6650 - val_loss: 0.6208
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.6788 - loss: 0.6217 - val_accuracy: 0.6680 - val_loss: 0.6237
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 92ms/step - accuracy: 0.6551 - loss: 0.6331 - val_accuracy: 0.6850 - val_loss: 0.5943
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.6843 - loss: 0.6111 - val_accuracy: 0.6940 - val_loss: 0.5792
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 58ms/step - accuracy: 0.7047 - loss: 0.5946 - val_accuracy: 0.6600 - val_loss: 0.6105
Epoch 14/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.6990 - loss: 0.5899 - val_accuracy: 0.7170 - val_loss: 0.5590
Epoch 15/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.7272 - loss: 0.5610 - val_accuracy: 0.6720 - val_loss: 0.6471
Epoch 16/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 54ms/step - accuracy: 0.6812 - loss: 0.5994 - val_accuracy: 0.7050 - val_loss: 0.5578
Epoch 17/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.7403 - loss: 0.5518 - val_accuracy: 0.7240 - val_loss: 0.5429
Epoch 18/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.7060 - loss: 0.5812 - val_accuracy: 0.7100 - val_loss: 0.5534
Epoch 19/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - accuracy: 0.7136 - loss: 0.5625 - val_accuracy: 0.7350 - val_loss: 0.5252
Epoch 20/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 60ms/step - accuracy: 0.7367 - loss: 0.5382 - val_accuracy: 0.7320 - val_loss: 0.5383
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.7167 - loss: 0.5890
Test Accuracy (Scratch CNN, 1500 samples): 0.728
```



=== Step 4: Pretrained VGG16 with Different Sample Sizes ===

```
# === Step 4: Pretrained VGG16 with Different Sample Sizes ===
def train_pretrained_vgg16(train_dataset, val_dataset, model_label):
    print(f"\n🚀 Starting training for {model_label}...")

    # Augmentation
    aug_layer = keras.Sequential([
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.15),
```

```
        layers.RandomZoom(0.25),
    ])

    # Load VGG16 base
    vgg_base = keras.applications.vgg16.VGG16(
        weights="imagenet",
        include_top=False,
        input_shape=(160, 160, 3)
    )
    vgg_base.trainable = False  # Freeze the base

    # Build model
    inputs = keras.Input(shape=(160, 160, 3))
    x = aug_layer(inputs)
    x = keras.applications.vgg16.preprocess_input(x)
    x = vgg_base(x)
    x = layers.Flatten()(x)
    x = layers.Dense(512, activation="relu")(x)  # Different dense layer size
    x = layers.Dropout(0.4)(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    vgg_model = keras.Model(inputs, outputs)

    # Compile with custom settings
    vgg_model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=2e-5),  # Different optimizer and rate
        loss="binary_crossentropy",
        metrics=["accuracy"]
    )

    # Add early stopping
    callbacks_list = [
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=4, restore_best_weights=True)
    ]

    # Train
    history = vgg_model.fit(train_dataset, epochs=20, validation_data=val_dataset, callbacks=callbacks_list)

    # Evaluate
    loss, acc = vgg_model.evaluate(test_data)
    print(f"{model_label} - Test Accuracy: {acc:.3f}")
    return history

# VGG16 with 1000 samples
train_data_vgg_1000 = complete_train_data.take(32)
history_vgg_1000 = train_pretrained_vgg16(train_data_vgg_1000, val_data, "VGG16 - 1000 Samples")

# VGG16 with 1500 samples
train_data_vgg_1500 = complete_train_data.take(47)
history_vgg_1500 = train_pretrained_vgg16(train_data_vgg_1500, val_data, "VGG16 - 1500 Samples")

# VGG16 with 2000 samples
train_data_vgg_2000 = complete_train_data.take(63)
history_vgg_2000 = train_pretrained_vgg16(train_data_vgg_2000, val_data, "VGG16 - 2000 Samples")

# Plot VGG16 Results
def plot_vgg_history(history, title, filename):
    train_acc = history.history["accuracy"]
    val_acc = history.history["val_accuracy"]
    plt.figure()
    plt.plot(train_acc, "bo-", label="Training Accuracy")
    plt.plot(val_acc, "ro-", label="Validation Accuracy")
    plt.title(title)
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.legend()
    plt.grid(True)
    plt.savefig(filename)
    plt.show()

plot_vgg_history(history_vgg_1000, "VGG16 (1000 Samples)", "vgg16_1000.png")
```

```
🚀 Starting training for VGG16 - 1000 Samples...
Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 169ms/step - accuracy: 0.5701 - loss: 5.4974 - val_accuracy: 0.8840 - val_loss: 1.0148
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 158ms/step - accuracy: 0.7832 - loss: 2.2469 - val_accuracy: 0.9490 - val_loss: 0.5668
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 160ms/step - accuracy: 0.8518 - loss: 1.3633 - val_accuracy: 0.9590 - val_loss: 0.4930
Epoch 4/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 160ms/step - accuracy: 0.8511 - loss: 1.3909 - val_accuracy: 0.9630 - val_loss: 0.4414
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 152ms/step - accuracy: 0.9162 - loss: 0.6080 - val_accuracy: 0.9560 - val_loss: 0.4297
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 155ms/step - accuracy: 0.8784 - loss: 1.0603 - val_accuracy: 0.9640 - val_loss: 0.4156
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 157ms/step - accuracy: 0.9055 - loss: 0.7739 - val_accuracy: 0.9630 - val_loss: 0.4225
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 156ms/step - accuracy: 0.9287 - loss: 0.6925 - val_accuracy: 0.9580 - val_loss: 0.4289
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 147ms/step - accuracy: 0.9339 - loss: 0.4364 - val_accuracy: 0.9630 - val_loss: 0.4123
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 157ms/step - accuracy: 0.9355 - loss: 0.3973 - val_accuracy: 0.9610 - val_loss: 0.4106
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 158ms/step - accuracy: 0.9264 - loss: 0.5930 - val_accuracy: 0.9610 - val_loss: 0.3964
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 141ms/step - accuracy: 0.9302 - loss: 0.4955 - val_accuracy: 0.9630 - val_loss: 0.4271
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 156ms/step - accuracy: 0.9384 - loss: 0.3929 - val_accuracy: 0.9610 - val_loss: 0.3907
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 147ms/step - accuracy: 0.9466 - loss: 0.5258 - val_accuracy: 0.9640 - val_loss: 0.3735
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 156ms/step - accuracy: 0.9257 - loss: 0.5317 - val_accuracy: 0.9630 - val_loss: 0.3688
Epoch 16/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 156ms/step - accuracy: 0.9244 - loss: 0.6725 - val_accuracy: 0.9640 - val_loss: 0.3466
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 148ms/step - accuracy: 0.9431 - loss: 0.3741 - val_accuracy: 0.9680 - val_loss: 0.3358
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 144ms/step - accuracy: 0.9570 - loss: 0.3290 - val_accuracy: 0.9670 - val_loss: 0.3418
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 156ms/step - accuracy: 0.9441 - loss: 0.4172 - val_accuracy: 0.9650 - val_loss: 0.3475
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 148ms/step - accuracy: 0.9595 - loss: 0.2781 - val_accuracy: 0.9630 - val_loss: 0.3733
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 70ms/step - accuracy: 0.9690 - loss: 0.3972
VGG16 - 1000 Samples - Test Accuracy: 0.968

🚀 Starting training for VGG16 - 1500 Samples...
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 135ms/step - accuracy: 0.6121 - loss: 4.8848 - val_accuracy: 0.9330 - val_loss: 0.5903
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 129ms/step - accuracy: 0.8558 - loss: 1.4480 - val_accuracy: 0.9460 - val_loss: 0.4890
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 130ms/step - accuracy: 0.8843 - loss: 1.2167 - val_accuracy: 0.9570 - val_loss: 0.4001
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 122ms/step - accuracy: 0.9011 - loss: 0.9414 - val_accuracy: 0.9620 - val_loss: 0.3569
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 122ms/step - accuracy: 0.9121 - loss: 1.0094 - val_accuracy: 0.9600 - val_loss: 0.3307
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 121ms/step - accuracy: 0.9198 - loss: 0.7076 - val_accuracy: 0.9700 - val_loss: 0.2763
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 123ms/step - accuracy: 0.9291 - loss: 0.7258 - val_accuracy: 0.9640 - val_loss: 0.2381
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 121ms/step - accuracy: 0.9239 - loss: 0.6706 - val_accuracy: 0.9680 - val_loss: 0.3103
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 124ms/step - accuracy: 0.9390 - loss: 0.5490 - val_accuracy: 0.9730 - val_loss: 0.2547
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 123ms/step - accuracy: 0.9384 - loss: 0.4792 - val_accuracy: 0.9730 - val_loss: 0.2251
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 124ms/step - accuracy: 0.9398 - loss: 0.4418 - val_accuracy: 0.9720 - val_loss: 0.2168
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 123ms/step - accuracy: 0.9249 - loss: 0.5041 - val_accuracy: 0.9750 - val_loss: 0.2437
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 129ms/step - accuracy: 0.9352 - loss: 0.4177 - val_accuracy: 0.9770 - val_loss: 0.2287
Epoch 14/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 123ms/step - accuracy: 0.9433 - loss: 0.3578 - val_accuracy: 0.9760 - val_loss: 0.2244
Epoch 15/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 120ms/step - accuracy: 0.9607 - loss: 0.4349 - val_accuracy: 0.9770 - val_loss: 0.2380
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.9754 - loss: 0.2774
VGG16 - 1500 Samples - Test Accuracy: 0.973

🚀 Starting training for VGG16 - 2000 Samples...
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.6652 - loss: 4.1595 - val_accuracy: 0.9390 - val_loss: 0.5920
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.8532 - loss: 1.6512 - val_accuracy: 0.9670 - val_loss: 0.3255
```

```
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.8779 - loss: 1.3111 - val_accuracy: 0.9680 - val_loss: 0.2471
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.9063 - loss: 0.8531 - val_accuracy: 0.9670 - val_loss: 0.3839
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 108ms/step - accuracy: 0.9101 - loss: 0.7455 - val_accuracy: 0.9730 - val_loss: 0.3025
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.9174 - loss: 0.7416 - val_accuracy: 0.9740 - val_loss: 0.2746
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.9263 - loss: 0.6120 - val_accuracy: 0.9700 - val_loss: 0.2670
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 70ms/step - accuracy: 0.9637 - loss: 0.3655
VGG16 - 2000 Samples - Test Accuracy: 0.960
```