```
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/test/cats/1870.jpg
  inflating: cats_vs_dogs_small/test/cats/1871.jpg
  inflating: cats_vs_dogs_small/test/cats/1872.jpg
  inflating: cats_vs_dogs_small/test/cats/1873.jpg
  inflating: cats_vs_dogs_small/test/cats/1874.jpg
  inflating: cats_vs_dogs_small/test/cats/1875.jpg
  inflating: cats_vs_dogs_small/test/cats/1876.jpg
  inflating: cats_vs_dogs_small/test/cats/1877.jpg
  inflating: cats_vs_dogs_small/test/cats/1878.jpg
  inflating: cats_vs_dogs_small/test/cats/1879.jpg
  inflating: cats_vs_dogs_small/test/cats/1880.jpg
  inflating: cats_vs_dogs_small/test/cats/1881.jpg
  inflating: cats_vs_dogs_small/test/cats/1882.jpg
  inflating: cats_vs_dogs_small/test/cats/1883.jpg
  inflating: cats_vs_dogs_small/test/cats/1884.jpg
  inflating: cats_vs_dogs_small/test/cats/1885.jpg
  inflating: cats_vs_dogs_small/test/cats/1886.jpg
  inflating: cats_vs_dogs_small/test/cats/1887.jpg
  inflating: cats_vs_dogs_small/test/cats/1888.jpg
  inflating: cats_vs_dogs_small/test/cats/1889.jpg
  inflating: cats_vs_dogs_small/test/cats/1890.jpg
  inflating: cats_vs_dogs_small/test/cats/1891.jpg
  inflating: cats_vs_dogs_small/test/cats/1892.jpg
  inflating: cats_vs_dogs_small/test/cats/1893.jpg
  inflating: cats_vs_dogs_small/test/cats/1894.jpg
  inflating: cats_vs_dogs_small/test/cats/1895.jpg
  inflating: cats_vs_dogs_small/test/cats/1896.jpg
  inflating: cats_vs_dogs_small/test/cats/1897.jpg
  inflating: cats_vs_dogs_small/test/cats/1898.jpg
  inflating: cats_vs_dogs_small/test/cats/1899.jpg
  inflating: cats_vs_dogs_small/test/cats/1900.jpg
  inflating: cats_vs_dogs_small/test/cats/1901.jpg
  inflating: cats_vs_dogs_small/test/cats/1902.jpg
  inflating: cats_vs_dogs_small/test/cats/1903.jpg
  inflating: cats_vs_dogs_small/test/cats/1904.jpg
  inflating: cats_vs_dogs_small/test/cats/1905.jpg
  inflating: cats_vs_dogs_small/test/cats/1906.jpg
  inflating: cats_vs_dogs_small/test/cats/1907.jpg
  inflating: cats_vs_dogs_small/test/cats/1908.jpg
  inflating: cats_vs_dogs_small/test/cats/1909.jpg
  inflating: cats_vs_dogs_small/test/cats/1910.jpg
  inflating: cats_vs_dogs_small/test/cats/1911.jpg
  inflating: cats_vs_dogs_small/test/cats/1912.jpg
  inflating: cats_vs_dogs_small/test/cats/1913.jpg
  inflating: cats_vs_dogs_small/test/cats/1914.jpg
  inflating: cats_vs_dogs_small/test/cats/1915.jpg
  inflating: cats_vs_dogs_small/test/cats/1916.jpg
  inflating: cats_vs_dogs_small/test/cats/1917.jpg
  inflating: cats_vs_dogs_small/test/cats/1918.jpg
  inflating: cats_vs_dogs_small/test/cats/1919.jpg
  inflating: cats_vs_dogs_small/test/cats/1920.jpg
  inflating: cats_vs_dogs_small/test/cats/1921.jpg
  inflating: cats_vs_dogs_small/test/cats/1922.jpg
 extracting: cats_vs_dogs_small/test/cats/1923.jpg
  inflating: cats_vs_dogs_small/test/cats/1924.jpg
  inflating: cats_vs_dogs_small/test/cats/1925.jpg
  inflating: cats_vs_dogs_small/test/cats/1926.jpg
  inflating: cats_vs_dogs_small/test/cats/1927.jpg
  inflating: cats_vs_dogs_small/test/cats/1928.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


base_dir = "/content/cats_vs_dogs_small"

full_train_ds = image_dataset_from_directory(
    os.path.join(base_dir, "train"),
    image_size=(180, 180),
    batch_size=32,
    shuffle=True,
    seed=123
)

validation_ds = image_dataset_from_directory(
    os.path.join(base_dir, "validation"),
    image_size=(180, 180),
    batch_size=32
)

test_ds = image_dataset_from_directory(
    os.path.join(base_dir, "test"),
    image_size=(180, 180),
    batch_size=32
)
```

```
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

CNN from Scratch – 1000 Samples

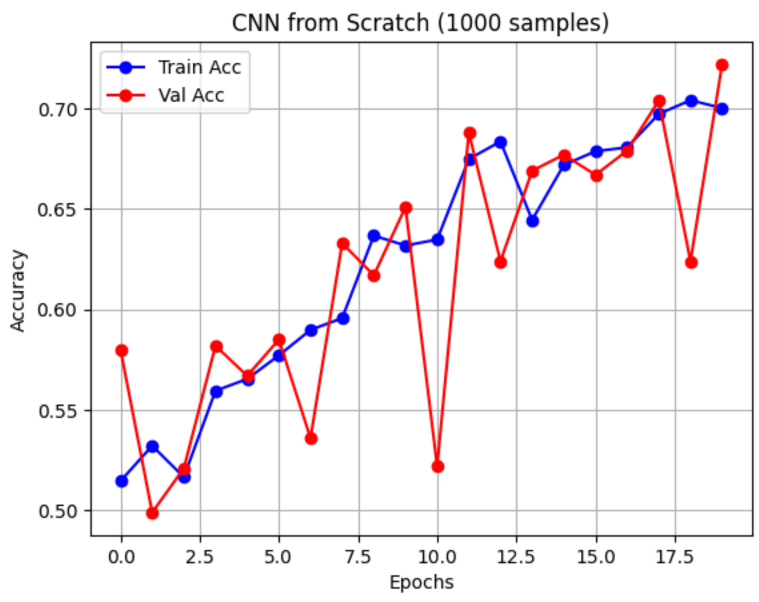
```
train_ds = full_train_ds.take(32)  # ~1024 samples

model = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    layers.Rescaling(1./255),
    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(128, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(256, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

history = model.fit(train_ds, epochs=20, validation_data=validation_ds)
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test Accuracy CNN 1000 samples: {test_acc:.3f}")
```

```
Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 84ms/step - accuracy: 0.5069 - loss: 0.7778 - val_accuracy: 0.5800 - val_loss: 0.6923
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 85ms/step - accuracy: 0.5254 - loss: 0.6940 - val_accuracy: 0.4990 - val_loss: 0.7011
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 133ms/step - accuracy: 0.5179 - loss: 0.6975 - val_accuracy: 0.5210 - val_loss: 0.6896
Epoch 4/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.5649 - loss: 0.6878 - val_accuracy: 0.5820 - val_loss: 0.6805
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.5501 - loss: 0.6816 - val_accuracy: 0.5670 - val_loss: 0.6785
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.5863 - loss: 0.6668 - val_accuracy: 0.5850 - val_loss: 0.6771
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 82ms/step - accuracy: 0.5630 - loss: 0.6883 - val_accuracy: 0.5360 - val_loss: 0.6905
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 136ms/step - accuracy: 0.5816 - loss: 0.6949 - val_accuracy: 0.6330 - val_loss: 0.6589
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 82ms/step - accuracy: 0.6258 - loss: 0.6536 - val_accuracy: 0.6170 - val_loss: 0.6348
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.6254 - loss: 0.6330 - val_accuracy: 0.6510 - val_loss: 0.6203
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 77ms/step - accuracy: 0.6411 - loss: 0.6385 - val_accuracy: 0.5220 - val_loss: 0.9989
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 77ms/step - accuracy: 0.6493 - loss: 0.6494 - val_accuracy: 0.6880 - val_loss: 0.5863
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 105ms/step - accuracy: 0.6796 - loss: 0.6188 - val_accuracy: 0.6240 - val_loss: 0.6269
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.6305 - loss: 0.6309 - val_accuracy: 0.6690 - val_loss: 0.6002
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.6460 - loss: 0.6220 - val_accuracy: 0.6770 - val_loss: 0.5848
Epoch 16/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 77ms/step - accuracy: 0.6802 - loss: 0.5933 - val_accuracy: 0.6670 - val_loss: 0.6397
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.6666 - loss: 0.6306 - val_accuracy: 0.6790 - val_loss: 0.5915
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 96ms/step - accuracy: 0.6905 - loss: 0.5830 - val_accuracy: 0.7040 - val_loss: 0.5649
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step - accuracy: 0.7087 - loss: 0.5706 - val_accuracy: 0.6240 - val_loss: 0.6582
```

```
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.7114 - loss: 0.5721 - val_accuracy: 0.7220 - val_loss: 0.5416
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.7175 - loss: 0.5742
Test Accuracy CNN 1000 samples: 0.718
```

Plot CNN Accuracy (1000 samples)

```
plt.plot(history.history['accuracy'], 'bo-', label="Train Acc")
plt.plot(history.history['val_accuracy'], 'ro-', label="Val Acc")
plt.title("CNN from Scratch (1000 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



CNN - 1500 Samples

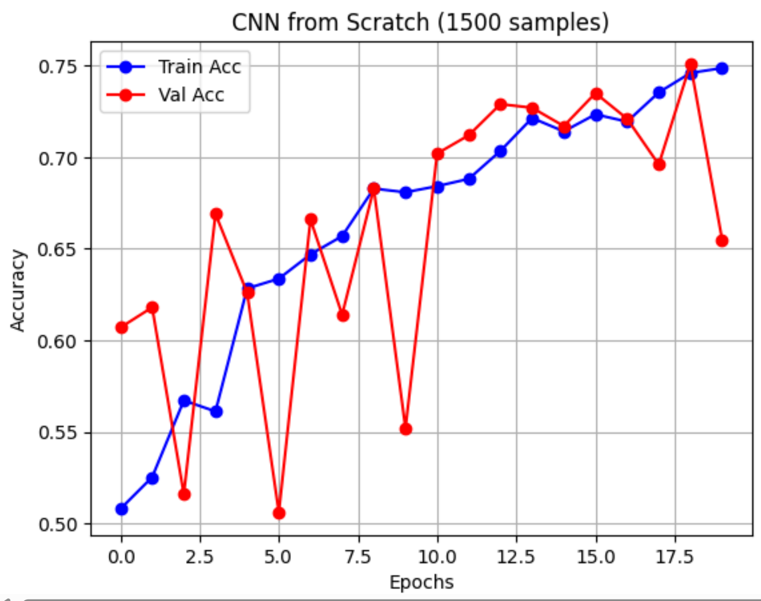
```
train_ds = full_train_ds.take(47)  # ~1500 samples

model = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    layers.Rescaling(1./255),
    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(128, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(256, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer="rmsprop", loss="binary_crossentropy", metrics=["accuracy"])
history = model.fit(train_ds, epochs=20, validation_data=validation_ds)
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test Accuracy CNN 1500 samples: {test_acc:.3f}")
```

```
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - accuracy: 0.4901 - loss: 0.7324 - val_accuracy: 0.6070 - val_loss: 0.6917
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 89ms/step - accuracy: 0.5212 - loss: 0.6923 - val_accuracy: 0.6180 - val_loss: 0.6899
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 56ms/step - accuracy: 0.5771 - loss: 0.6927 - val_accuracy: 0.5160 - val_loss: 0.6832
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 57ms/step - accuracy: 0.5455 - loss: 0.6887 - val_accuracy: 0.6690 - val_loss: 0.6462
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 56ms/step - accuracy: 0.6303 - loss: 0.6587 - val_accuracy: 0.6260 - val_loss: 0.6337
```

```
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - accuracy: 0.6110 - loss: 0.6578 - val_accuracy: 0.5060 - val_loss: 0.9313
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 58ms/step - accuracy: 0.6277 - loss: 0.6649 - val_accuracy: 0.6660 - val_loss: 0.5998
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.6707 - loss: 0.6171 - val_accuracy: 0.6140 - val_loss: 0.6534
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 57ms/step - accuracy: 0.6833 - loss: 0.6103 - val_accuracy: 0.6830 - val_loss: 0.5921
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 85ms/step - accuracy: 0.6831 - loss: 0.6001 - val_accuracy: 0.5520 - val_loss: 0.7240
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 66ms/step - accuracy: 0.6696 - loss: 0.6172 - val_accuracy: 0.7020 - val_loss: 0.5693
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 5s 65ms/step - accuracy: 0.6954 - loss: 0.6057 - val_accuracy: 0.7120 - val_loss: 0.5622
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 92ms/step - accuracy: 0.6845 - loss: 0.5926 - val_accuracy: 0.7290 - val_loss: 0.5259
Epoch 14/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.7245 - loss: 0.5751 - val_accuracy: 0.7270 - val_loss: 0.5330
Epoch 15/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.7150 - loss: 0.5669 - val_accuracy: 0.7170 - val_loss: 0.5571
Epoch 16/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.7128 - loss: 0.5717 - val_accuracy: 0.7350 - val_loss: 0.5239
Epoch 17/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 4s 96ms/step - accuracy: 0.7122 - loss: 0.5438 - val_accuracy: 0.7210 - val_loss: 0.5229
Epoch 18/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - accuracy: 0.7388 - loss: 0.5376 - val_accuracy: 0.6960 - val_loss: 0.5677
Epoch 19/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.7391 - loss: 0.5257 - val_accuracy: 0.7510 - val_loss: 0.5119
Epoch 20/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.7468 - loss: 0.5135 - val_accuracy: 0.6550 - val_loss: 0.6650
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.6653 - loss: 0.6750
Test Accuracy CNN 1500 samples: 0.651
```

Plot CNN Accuracy (1500 samples)

```
plt.plot(history.history['accuracy'], 'bo-', label="Train Acc")
plt.plot(history.history['val_accuracy'], 'ro-', label="Val Acc")
plt.title("CNN from Scratch (1500 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



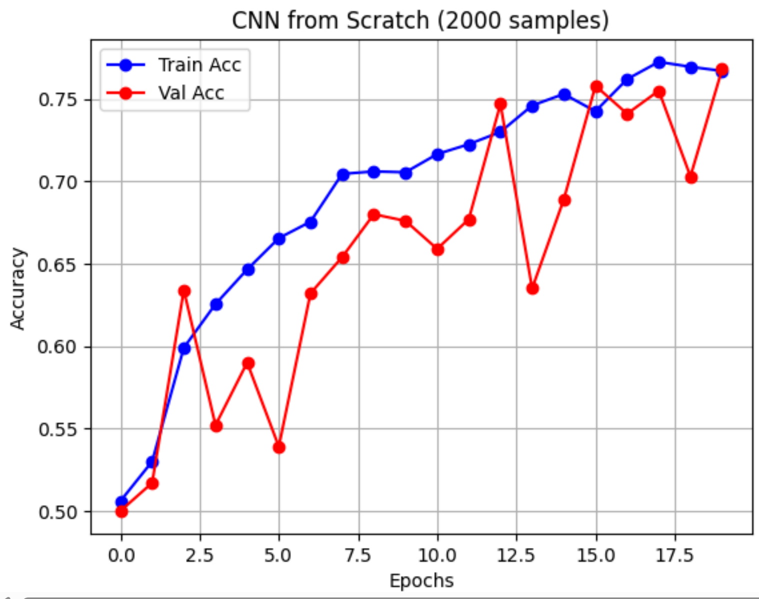
CNN - 2000 Samples

```
train_ds = full_train_ds.take(63)  # ~2000 samples

model = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    layers.Rescaling(1./255),
    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(128, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Conv2D(256, 3, activation='relu'),
    layers.MaxPooling2D(2),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer="rmsprop", loss="binary_crossentropy", metrics=["accuracy"])
history = model.fit(train_ds, epochs=20, validation_data=validation_ds)
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test Accuracy CNN 2000 samples: {test_acc:.3f}")
```

```
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 67ms/step - accuracy: 0.5111 - loss: 0.7371 - val_accuracy: 0.5000 - val_loss: 0.7425
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.5143 - loss: 0.6967 - val_accuracy: 0.5170 - val_loss: 0.6839
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 82ms/step - accuracy: 0.6010 - loss: 0.6983 - val_accuracy: 0.6340 - val_loss: 0.6391
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.5995 - loss: 0.6650 - val_accuracy: 0.5520 - val_loss: 0.6922
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 87ms/step - accuracy: 0.6371 - loss: 0.6364 - val_accuracy: 0.5900 - val_loss: 0.6725
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.6486 - loss: 0.6478 - val_accuracy: 0.5390 - val_loss: 0.8473
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.6578 - loss: 0.6266 - val_accuracy: 0.6320 - val_loss: 0.6553
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 79ms/step - accuracy: 0.7030 - loss: 0.5962 - val_accuracy: 0.6540 - val_loss: 0.6096
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 63ms/step - accuracy: 0.7013 - loss: 0.5846 - val_accuracy: 0.6800 - val_loss: 0.6029
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.6971 - loss: 0.5891 - val_accuracy: 0.6760 - val_loss: 0.6510
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.7060 - loss: 0.5637 - val_accuracy: 0.6590 - val_loss: 0.6743
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 63ms/step - accuracy: 0.7220 - loss: 0.5526 - val_accuracy: 0.6770 - val_loss: 0.6266
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.7230 - loss: 0.5557 - val_accuracy: 0.7470 - val_loss: 0.5027
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 86ms/step - accuracy: 0.7404 - loss: 0.5367 - val_accuracy: 0.6350 - val_loss: 0.7040
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.7303 - loss: 0.5489 - val_accuracy: 0.6890 - val_loss: 0.6280
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.7420 - loss: 0.5187 - val_accuracy: 0.7580 - val_loss: 0.5163
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 58ms/step - accuracy: 0.7574 - loss: 0.5115 - val_accuracy: 0.7410 - val_loss: 0.4989
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 74ms/step - accuracy: 0.7743 - loss: 0.4996 - val_accuracy: 0.7550 - val_loss: 0.5060
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.7629 - loss: 0.5046 - val_accuracy: 0.7030 - val_loss: 0.6150
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.7528 - loss: 0.5131 - val_accuracy: 0.7680 - val_loss: 0.4855
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.7642 - loss: 0.5105
Test Accuracy CNN 2000 samples: 0.744
```

Plot CNN Accuracy (2000 samples)

```
plt.plot(history.history['accuracy'], 'bo-', label="Train Acc")
plt.plot(history.history['val_accuracy'], 'ro-', label="Val Acc")
plt.title("CNN from Scratch (2000 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



VGG16 – 1000 Samples

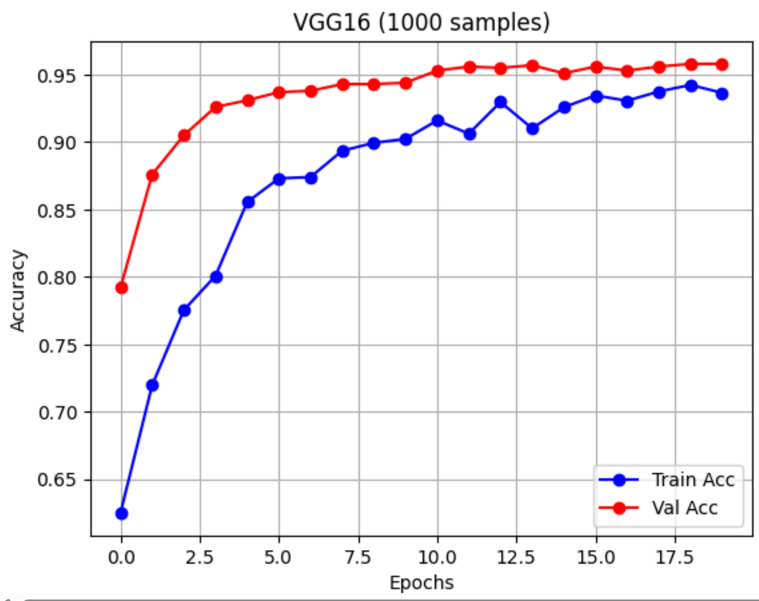
```
train_ds = full_train_ds.take(32)  # ~1000 samples

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

# Load base model
conv_base = keras.applications.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3)
)
conv_base.trainable = False  # Freeze pretrained layers

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

# Compile model with learning rate tuning
model.compile(
    optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

# Early stopping
callbacks = [
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True)
]

# Train
history = model.fit(train_ds, epochs=20, validation_data=validation_ds, callbacks=callbacks)

# Evaluate
test_loss, test_acc = model.evaluate(test_ds)
print(f"VGG16 (1000 samples) Test Accuracy: {test_acc:.3f}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_no
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 234ms/step - accuracy: 0.5824 - loss: 6.6830 - val_accuracy: 0.7920 - val_loss: 1.9963
```

```
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 189ms/step - accuracy: 0.6816 - loss: 3.6027 - val_accuracy: 0.8760 - val_loss: 1.2569
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 192ms/step - accuracy: 0.7632 - loss: 2.5682 - val_accuracy: 0.9050 - val_loss: 0.9847
Epoch 4/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 198ms/step - accuracy: 0.8025 - loss: 2.1565 - val_accuracy: 0.9260 - val_loss: 0.8253
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 198ms/step - accuracy: 0.8530 - loss: 1.5135 - val_accuracy: 0.9310 - val_loss: 0.7369
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 262ms/step - accuracy: 0.8859 - loss: 1.2784 - val_accuracy: 0.9370 - val_loss: 0.6700
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 190ms/step - accuracy: 0.8627 - loss: 1.7421 - val_accuracy: 0.9380 - val_loss: 0.6304
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 196ms/step - accuracy: 0.8990 - loss: 1.0367 - val_accuracy: 0.9430 - val_loss: 0.5870
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 260ms/step - accuracy: 0.9221 - loss: 0.6831 - val_accuracy: 0.9430 - val_loss: 0.5704
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 261ms/step - accuracy: 0.9009 - loss: 0.9756 - val_accuracy: 0.9440 - val_loss: 0.5426
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 195ms/step - accuracy: 0.9038 - loss: 0.8315 - val_accuracy: 0.9530 - val_loss: 0.5280
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 262ms/step - accuracy: 0.9042 - loss: 0.9058 - val_accuracy: 0.9560 - val_loss: 0.4974
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 195ms/step - accuracy: 0.9302 - loss: 0.7148 - val_accuracy: 0.9550 - val_loss: 0.4912
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 263ms/step - accuracy: 0.9146 - loss: 0.9284 - val_accuracy: 0.9570 - val_loss: 0.4790
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 193ms/step - accuracy: 0.9299 - loss: 0.7432 - val_accuracy: 0.9510 - val_loss: 0.4924
Epoch 16/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 192ms/step - accuracy: 0.9286 - loss: 0.6966 - val_accuracy: 0.9560 - val_loss: 0.4622
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 191ms/step - accuracy: 0.9392 - loss: 0.5484 - val_accuracy: 0.9530 - val_loss: 0.4622
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 267ms/step - accuracy: 0.9347 - loss: 0.7279 - val_accuracy: 0.9560 - val_loss: 0.4430
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 260ms/step - accuracy: 0.9537 - loss: 0.4489 - val_accuracy: 0.9580 - val_loss: 0.4335
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 260ms/step - accuracy: 0.9373 - loss: 0.5928 - val_accuracy: 0.9580 - val_loss: 0.4201
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 91ms/step - accuracy: 0.9676 - loss: 0.2875
VGG16 (1000 samples) Test Accuracy: 0.964
```

Plot Accuracy for 1000 Samples

```
plt.plot(history.history['accuracy'], 'bo-', label='Train Acc')
plt.plot(history.history['val_accuracy'], 'ro-', label='Val Acc')
plt.title("VGG16 (1000 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```

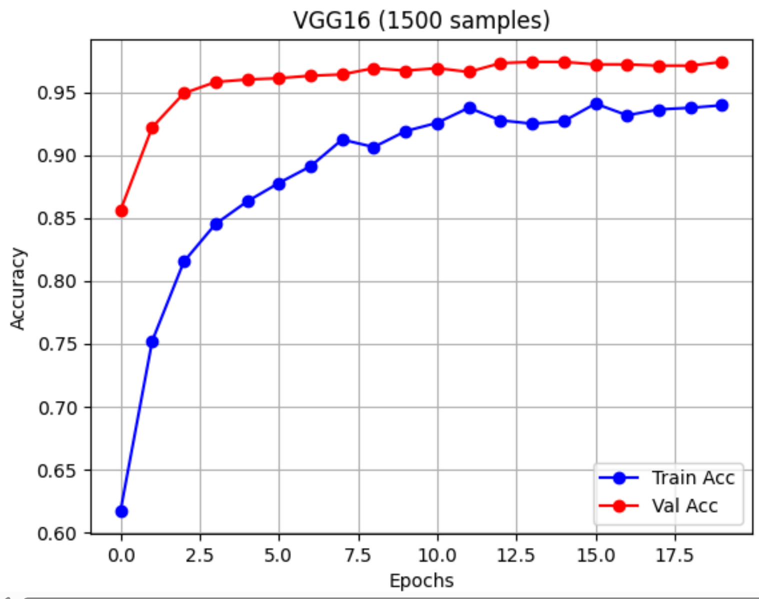


VGG16 – 1500 Samples

```
train_ds = full_train_ds.take(47)  # ~1500 samples

conv_base.trainable = False  # Re-freeze if needed

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

model.compile(optimizer=keras.optimizers.RMSprop(1e-5),
              loss="binary_crossentropy",
              metrics=["accuracy"])

callbacks = [
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True)
]

history = model.fit(train_ds, epochs=20, validation_data=validation_ds, callbacks=callbacks)

test_loss, test_acc = model.evaluate(test_ds)
print(f"VGG16 (1500 samples) Test Accuracy: {test_acc:.3f}")
```

```
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 167ms/step - accuracy: 0.5466 - loss: 6.9906 - val_accuracy: 0.8560 - val_loss: 1.0636
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 209ms/step - accuracy: 0.7544 - loss: 3.1172 - val_accuracy: 0.9220 - val_loss: 0.5192
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 163ms/step - accuracy: 0.8027 - loss: 1.9083 - val_accuracy: 0.9490 - val_loss: 0.3747
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 163ms/step - accuracy: 0.8415 - loss: 1.5422 - val_accuracy: 0.9580 - val_loss: 0.3101
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 162ms/step - accuracy: 0.8482 - loss: 1.4130 - val_accuracy: 0.9600 - val_loss: 0.2701
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 207ms/step - accuracy: 0.8761 - loss: 1.2812 - val_accuracy: 0.9610 - val_loss: 0.2565
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 159ms/step - accuracy: 0.8912 - loss: 1.2111 - val_accuracy: 0.9630 - val_loss: 0.2410
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 206ms/step - accuracy: 0.9117 - loss: 0.8664 - val_accuracy: 0.9640 - val_loss: 0.2250
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 206ms/step - accuracy: 0.9015 - loss: 0.8651 - val_accuracy: 0.9690 - val_loss: 0.2080
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 157ms/step - accuracy: 0.9136 - loss: 0.7236 - val_accuracy: 0.9670 - val_loss: 0.2170
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 160ms/step - accuracy: 0.9238 - loss: 0.6590 - val_accuracy: 0.9690 - val_loss: 0.2059
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 161ms/step - accuracy: 0.9371 - loss: 0.3689 - val_accuracy: 0.9660 - val_loss: 0.1988
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 161ms/step - accuracy: 0.9161 - loss: 0.6675 - val_accuracy: 0.9730 - val_loss: 0.2069
Epoch 14/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 208ms/step - accuracy: 0.9265 - loss: 0.5148 - val_accuracy: 0.9740 - val_loss: 0.1948
Epoch 15/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 206ms/step - accuracy: 0.9247 - loss: 0.5438 - val_accuracy: 0.9740 - val_loss: 0.1964
Epoch 16/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 157ms/step - accuracy: 0.9488 - loss: 0.3780 - val_accuracy: 0.9720 - val_loss: 0.1909
Epoch 17/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 158ms/step - accuracy: 0.9346 - loss: 0.4847 - val_accuracy: 0.9720 - val_loss: 0.2144
Epoch 18/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 158ms/step - accuracy: 0.9365 - loss: 0.5711 - val_accuracy: 0.9710 - val_loss: 0.1834
Epoch 19/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 159ms/step - accuracy: 0.9422 - loss: 0.4087 - val_accuracy: 0.9710 - val_loss: 0.1841
Epoch 20/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 208ms/step - accuracy: 0.9463 - loss: 0.3645 - val_accuracy: 0.9740 - val_loss: 0.1748
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 91ms/step - accuracy: 0.9649 - loss: 0.2906
VGG16 (1500 samples) Test Accuracy: 0.966
```

```
plt.plot(history.history['accuracy'], 'bo-', label='Train Acc')
plt.plot(history.history['val_accuracy'], 'ro-', label='Val Acc')
plt.title("VGG16 (1500 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



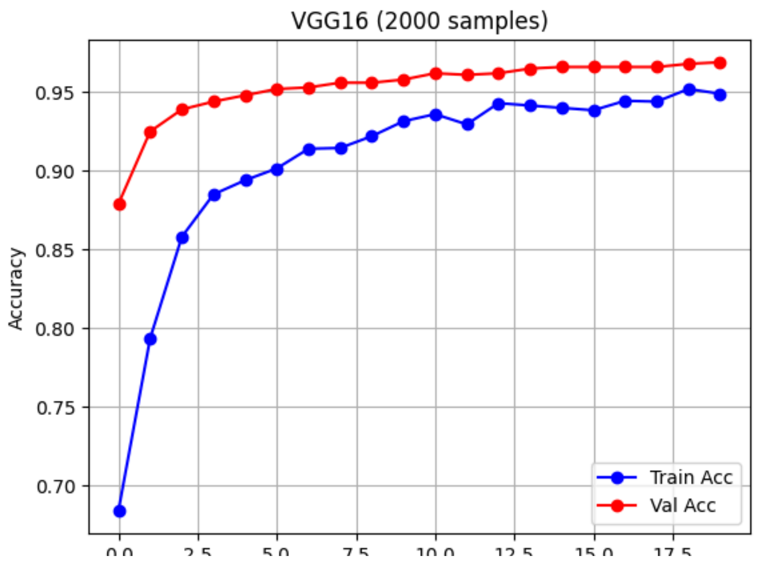
VGG16 – 2000 Samples

```
train_ds = full_train_ds.take(63)  # ~2000 samples

conv_base.trainable = False

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

model.compile(optimizer=keras.optimizers.RMSprop(1e-5),
              loss="binary_crossentropy",
              metrics=["accuracy"])

callbacks = [
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True)
]

history = model.fit(train_ds, epochs=20, validation_data=validation_ds, callbacks=callbacks)

test_loss, test_acc = model.evaluate(test_ds)
print(f"VGG16 (2000 samples) Test Accuracy: {test_acc:.3f}")
```

```
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 14s 202ms/step - accuracy: 0.6359 - loss: 5.1452 - val_accuracy: 0.8790 - val_loss: 1.1770
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 178ms/step - accuracy: 0.7695 - loss: 2.5916 - val_accuracy: 0.9250 - val_loss: 0.7692
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 142ms/step - accuracy: 0.8585 - loss: 1.5701 - val_accuracy: 0.9390 - val_loss: 0.6354
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 178ms/step - accuracy: 0.8844 - loss: 1.2081 - val_accuracy: 0.9440 - val_loss: 0.5564
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 142ms/step - accuracy: 0.8938 - loss: 1.1154 - val_accuracy: 0.9480 - val_loss: 0.5095
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 142ms/step - accuracy: 0.9076 - loss: 0.7909 - val_accuracy: 0.9520 - val_loss: 0.4702
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 141ms/step - accuracy: 0.9136 - loss: 0.9025 - val_accuracy: 0.9530 - val_loss: 0.4330
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 176ms/step - accuracy: 0.9243 - loss: 0.6931 - val_accuracy: 0.9560 - val_loss: 0.4073
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 139ms/step - accuracy: 0.9096 - loss: 0.7516 - val_accuracy: 0.9560 - val_loss: 0.3969
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 177ms/step - accuracy: 0.9324 - loss: 0.6195 - val_accuracy: 0.9580 - val_loss: 0.3740
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 178ms/step - accuracy: 0.9328 - loss: 0.4568 - val_accuracy: 0.9620 - val_loss: 0.3662
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 179ms/step - accuracy: 0.9257 - loss: 0.5597 - val_accuracy: 0.9610 - val_loss: 0.3567
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 145ms/step - accuracy: 0.9377 - loss: 0.5250 - val_accuracy: 0.9620 - val_loss: 0.3424
```

```
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 144ms/step - accuracy: 0.9447 - loss: 0.4458 - val_accuracy: 0.9650 - val_loss: 0.3325
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 178ms/step - accuracy: 0.9373 - loss: 0.4265 - val_accuracy: 0.9660 - val_loss: 0.3223
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 177ms/step - accuracy: 0.9398 - loss: 0.5313 - val_accuracy: 0.9660 - val_loss: 0.3059
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 175ms/step - accuracy: 0.9469 - loss: 0.3662 - val_accuracy: 0.9660 - val_loss: 0.2997
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 139ms/step - accuracy: 0.9363 - loss: 0.3887 - val_accuracy: 0.9660 - val_loss: 0.2908
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 142ms/step - accuracy: 0.9460 - loss: 0.3623 - val_accuracy: 0.9680 - val_loss: 0.2842
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 145ms/step - accuracy: 0.9494 - loss: 0.4591 - val_accuracy: 0.9690 - val_loss: 0.2790
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 0.9738 - loss: 0.1806
VGG16 (2000 samples) Test Accuracy: 0.974
```

```
plt.plot(history.history['accuracy'], 'bo-', label='Train Acc')
plt.plot(history.history['val_accuracy'], 'ro-', label='Val Acc')
plt.title("VGG16 (2000 samples)")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```


VGG16 (2000 samples)
Accuracy
Train Acc
Val Acc