Double-click (or enter) to edit

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

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

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import pandas as pd
import os

# Set the dataset path
base_dir = "/content/drive/MyDrive/cats_vs_dogs_small"

# Data Augmentation and Preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    os.path.join(base_dir, "train"),
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

validation_generator = val_test_datagen.flow_from_directory(
    os.path.join(base_dir, "validation"),
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(base_dir, "test"),
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

# Define CNN Model from Scratch
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
```

```
history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=20,
    validation_data=validation_generator,
    validation_steps=50
)

# Evaluate performance
test_loss, test_acc = model.evaluate(test_generator)
print(f"CNN from Scratch - Test Accuracy: {test_acc:.4f}")

# Transfer Learning with Pretrained Model
base_model = keras.applications.VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
base_model.trainable = False

model_tl = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

model_tl.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(learning_rate=0.0001), metrics=['accuracy'])

history_tl = model_tl.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=50
)

# Evaluate pretrained model
test_loss_tl, test_acc_tl = model_tl.evaluate(test_generator)
print(f"Pretrained VGG16 - Test Accuracy: {test_acc_tl:.4f}")

# Compare results
results = pd.DataFrame({
    'Model': ['CNN from Scratch', 'Pretrained VGG16'],
    'Test Accuracy': [test_acc, test_acc_tl]
})

print(results)

# Plot accuracy comparison
plt.figure(figsize=(8,5))
plt.bar(results['Model'], results['Test Accuracy'], color=['blue', 'green'])
plt.xlabel("Model")
plt.ylabel("Test Accuracy")
plt.title("Model Performance Comparison")
plt.show()
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyD
  self._warn_if_super_not_called()
Epoch 1/20
 63/100 ━━━━━━━━━━━━━━━━━━━━ 6:59 11s/step - accuracy: 0.5281 - loss: 0.7446/usr/local/lib/python3.11/dist-packages/keras/
  self._interrupted_warning()
100/100 ━━━━━━━━━━━━━━━━━━━━ 1441s 14s/step - accuracy: 0.5268 - loss: 0.7306 - val_accuracy: 0.5160 - val_loss: 0.6878
Epoch 2/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 46s 227ms/step - accuracy: 0.5171 - loss: 0.6909 - val_accuracy: 0.5790 - val_loss: 0.6852
Epoch 3/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 24s 241ms/step - accuracy: 0.5471 - loss: 0.6840 - val_accuracy: 0.5260 - val_loss: 0.6831
Epoch 4/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 40s 233ms/step - accuracy: 0.5501 - loss: 0.6829 - val_accuracy: 0.6010 - val_loss: 0.6757
Epoch 5/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 235ms/step - accuracy: 0.5927 - loss: 0.6744 - val_accuracy: 0.5960 - val_loss: 0.6588
Epoch 6/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 239ms/step - accuracy: 0.5870 - loss: 0.6727 - val_accuracy: 0.5440 - val_loss: 0.7240
Epoch 7/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 24s 236ms/step - accuracy: 0.5825 - loss: 0.6751 - val_accuracy: 0.6470 - val_loss: 0.6257
Epoch 8/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 23s 233ms/step - accuracy: 0.6290 - loss: 0.6557 - val_accuracy: 0.6110 - val_loss: 0.6623
Epoch 9/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 23s 228ms/step - accuracy: 0.6275 - loss: 0.6524 - val_accuracy: 0.5550 - val_loss: 0.7515
Epoch 10/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 42s 241ms/step - accuracy: 0.6367 - loss: 0.6537 - val_accuracy: 0.6780 - val_loss: 0.5988
Epoch 11/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 243ms/step - accuracy: 0.6614 - loss: 0.6066 - val_accuracy: 0.6690 - val_loss: 0.6143
Epoch 12/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 24s 237ms/step - accuracy: 0.6766 - loss: 0.5991 - val_accuracy: 0.7110 - val_loss: 0.5810
Epoch 13/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 234ms/step - accuracy: 0.7010 - loss: 0.5863 - val_accuracy: 0.6390 - val_loss: 0.6066
Epoch 14/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 23s 230ms/step - accuracy: 0.6707 - loss: 0.6083 - val_accuracy: 0.6970 - val_loss: 0.5719
Epoch 15/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 236ms/step - accuracy: 0.6847 - loss: 0.5992 - val_accuracy: 0.7110 - val_loss: 0.5448
Epoch 16/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 24s 236ms/step - accuracy: 0.6979 - loss: 0.5842 - val_accuracy: 0.7040 - val_loss: 0.5711
Epoch 17/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 23s 227ms/step - accuracy: 0.6898 - loss: 0.5874 - val_accuracy: 0.6970 - val_loss: 0.5809
Epoch 18/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 25s 248ms/step - accuracy: 0.7246 - loss: 0.5536 - val_accuracy: 0.7220 - val_loss: 0.5365
Epoch 19/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 40s 240ms/step - accuracy: 0.7341 - loss: 0.5403 - val_accuracy: 0.7120 - val_loss: 0.5511
Epoch 20/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 40s 232ms/step - accuracy: 0.7260 - loss: 0.5349 - val_accuracy: 0.7410 - val_loss: 0.5205
32/32 ━━━━━━━━━━━━━━━━━━━━ 674s 21s/step - accuracy: 0.7218 - loss: 0.5443
CNN from Scratch - Test Accuracy: 0.7160
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Epoch 1/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 44s 341ms/step - accuracy: 0.6253 - loss: 0.6318 - val_accuracy: 0.8580 - val_loss: 0.3607
Epoch 2/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 26s 256ms/step - accuracy: 0.7822 - loss: 0.4490 - val_accuracy: 0.8400 - val_loss: 0.3294
Epoch 3/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 252ms/step - accuracy: 0.8136 - loss: 0.4144 - val_accuracy: 0.8690 - val_loss: 0.3081
Epoch 4/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 25s 249ms/step - accuracy: 0.8259 - loss: 0.3869 - val_accuracy: 0.8850 - val_loss: 0.2765
Epoch 5/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 26s 257ms/step - accuracy: 0.8159 - loss: 0.3985 - val_accuracy: 0.8880 - val_loss: 0.2622
Epoch 6/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 26s 256ms/step - accuracy: 0.8264 - loss: 0.3784 - val_accuracy: 0.8890 - val_loss: 0.2593
Epoch 7/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 40s 247ms/step - accuracy: 0.8304 - loss: 0.3614 - val_accuracy: 0.8830 - val_loss: 0.2560
Epoch 8/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 26s 255ms/step - accuracy: 0.8457 - loss: 0.3484 - val_accuracy: 0.8810 - val_loss: 0.2664
Epoch 9/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 41s 253ms/step - accuracy: 0.8509 - loss: 0.3356 - val_accuracy: 0.8860 - val_loss: 0.2538
Epoch 10/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 25s 251ms/step - accuracy: 0.8590 - loss: 0.3301 - val_accuracy: 0.8880 - val_loss: 0.2482
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 141ms/step - accuracy: 0.8922 - loss: 0.2731
Pretrained VGG16 - Test Accuracy: 0.9000
              Model  Test Accuracy
0  CNN from Scratch          0.716
1  Pretrained VGG16          0.900
```