

```
!pip install -q kaggle
```

```
from google.colab import files
files.upload()
```

 Choose Files kaggle.json

- **kaggle.json**(application/json) - 74 bytes, last modified: 3/19/2025 - 100% done
Saving kaggle.json to kaggle.json

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d salader/dogs-vs-cats
```

 Dataset URL: <https://www.kaggle.com/datasets/salader/dogs-vs-cats>
License(s): unknown


```
import zipfile
zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

```
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Flatten
from keras.applications.vgg16 import VGG16
```

```
conv_base=VGG16(weights='imagenet',include_top = False,input_shape=(150,150,3))
```

 Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.58889256/58889256 ————— 2s 0us/step

```
conv_base.trainable = True
set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
for layer in conv_base.layers:
    print(layer.name, layer.trainable)
```

 input_layer False
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False
block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 False
block4_pool False
block5_conv1 True
block5_conv2 True
block5_conv3 True
block5_pool True

```
conv_base.summary()
```

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

```
model = Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

```
# generators
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/train',labels='inferred',
    label_mode = 'int',batch_size=32, image_size=(150,150))
```

```
validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/test',labels='inferred',
    label_mode = 'int',batch_size=32,image_size=(150,150))
```

Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.

```
# Normalize
def process(image,label):
    image = tensorflow.cast(image/255. ,tensorflow.float32)
    return image,label
```

```
train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)
```

```
model.compile(
    optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
    loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```

Epoch 1/10
625/625 ————— 97s 139ms/step - accuracy: 0.8561 - loss: 0.3209 - val_accuracy: 0.9312 - val_loss: 0.1673
Epoch 2/10
625/625 ————— 90s 143ms/step - accuracy: 0.9414 - loss: 0.1472 - val_accuracy: 0.9264 - val_loss: 0.1723
Epoch 3/10
625/625 ————— 86s 137ms/step - accuracy: 0.9596 - loss: 0.1035 - val_accuracy: 0.9416 - val_loss: 0.1522
Epoch 4/10
625/625 ————— 88s 140ms/step - accuracy: 0.9734 - loss: 0.0726 - val_accuracy: 0.9506 - val_loss: 0.1291
Epoch 5/10
625/625 ————— 145s 144ms/step - accuracy: 0.9847 - loss: 0.0497 - val_accuracy: 0.9486 - val_loss: 0.1270
Epoch 6/10
625/625 ————— 86s 138ms/step - accuracy: 0.9903 - loss: 0.0339 - val_accuracy: 0.9536 - val_loss: 0.1242
Epoch 7/10
625/625 ————— 87s 138ms/step - accuracy: 0.9946 - loss: 0.0216 - val_accuracy: 0.9394 - val_loss: 0.1886
Epoch 8/10
625/625 ————— 140s 136ms/step - accuracy: 0.9969 - loss: 0.0137 - val_accuracy: 0.9532 - val_loss: 0.1503
Epoch 9/10
625/625 ————— 141s 135ms/step - accuracy: 0.9982 - loss: 0.0094 - val_accuracy: 0.9488 - val_loss: 0.1788
Epoch 10/10
625/625 ————— 86s 138ms/step - accuracy: 0.9983 - loss: 0.0067 - val_accuracy: 0.9538 - val_loss: 0.1735

```

```
# Evaluate the model on the test data
```

```
loss, accuracy = model.evaluate(validation_ds)
```

```
print('Test accuracy:', accuracy)
```

```

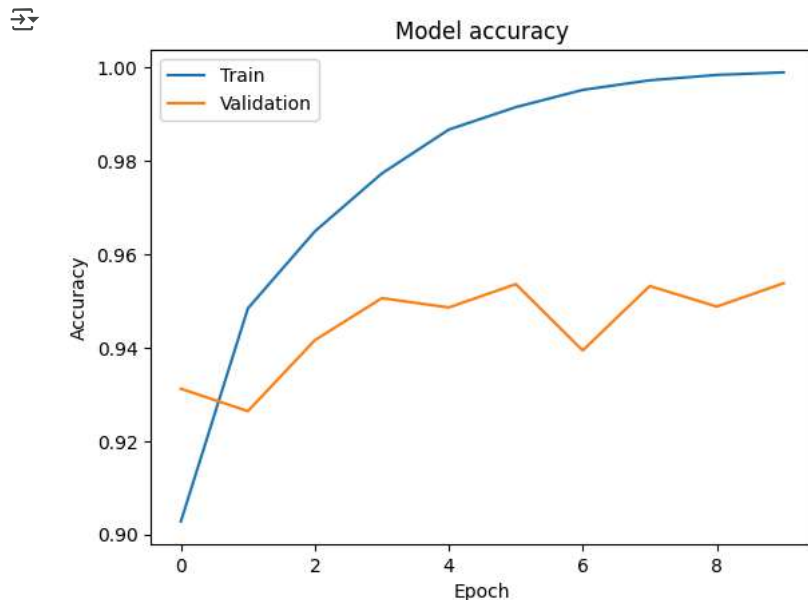
157/157 ————— 15s 93ms/step - accuracy: 0.9543 - loss: 0.1698
Test accuracy: 0.9538000226020813

```

```

import matplotlib.pyplot as plt
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

```



```

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

```

