```
!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
```

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

```
conv_base.trainable = False
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import array_to_img, img_to_array, load_img
```

```
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,
                zoom_range=0.2,horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory('/content/train',        target_size=(150, 150),batch_size=batch_size,

validation_generator = test_datagen.flow_from_directory('/content/test',target_size=(150, 150),batch_size=batch_size,class_
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
```

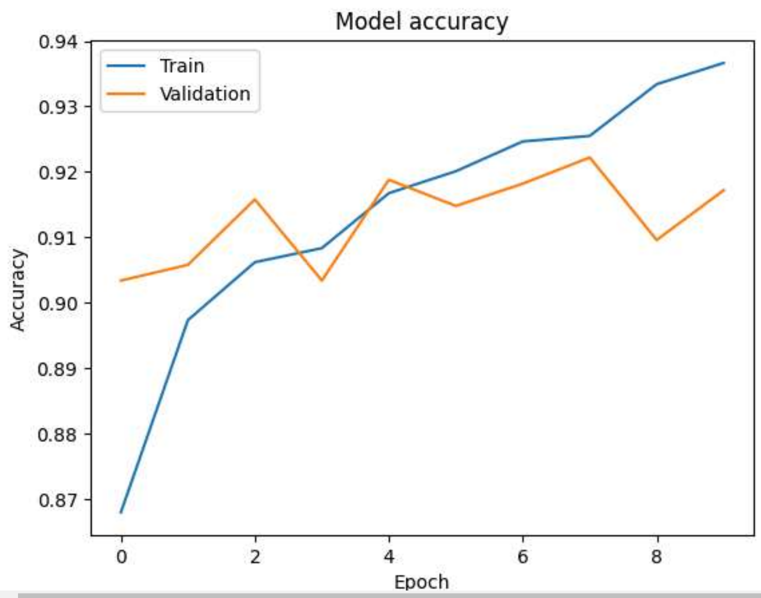
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
history = model.fit(train_generator,epochs=10,validation_data=validation_generator)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class
  self._warn_if_super_not_called()
Epoch 1/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 187s 285ms/step - accuracy: 0.8364 - loss: 0.3983 - val_accuracy: 0.9034 - val_loss: 0.2193
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 170s 271ms/step - accuracy: 0.8955 - loss: 0.2435 - val_accuracy: 0.9058 - val_loss: 0.2174
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 160s 257ms/step - accuracy: 0.9057 - loss: 0.2184 - val_accuracy: 0.9158 - val_loss: 0.1976
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 161s 258ms/step - accuracy: 0.9086 - loss: 0.2110 - val_accuracy: 0.9034 - val_loss: 0.2277
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 161s 258ms/step - accuracy: 0.9200 - loss: 0.1896 - val_accuracy: 0.9188 - val_loss: 0.1886
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 160s 255ms/step - accuracy: 0.9239 - loss: 0.1807 - val_accuracy: 0.9148 - val_loss: 0.1937
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 161s 257ms/step - accuracy: 0.9251 - loss: 0.1733 - val_accuracy: 0.9182 - val_loss: 0.1866
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 202s 257ms/step - accuracy: 0.9320 - loss: 0.1671 - val_accuracy: 0.9222 - val_loss: 0.1879
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 160s 256ms/step - accuracy: 0.9349 - loss: 0.1555 - val_accuracy: 0.9096 - val_loss: 0.2257
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 162s 259ms/step - accuracy: 0.9372 - loss: 0.1519 - val_accuracy: 0.9172 - val_loss: 0.1933
```

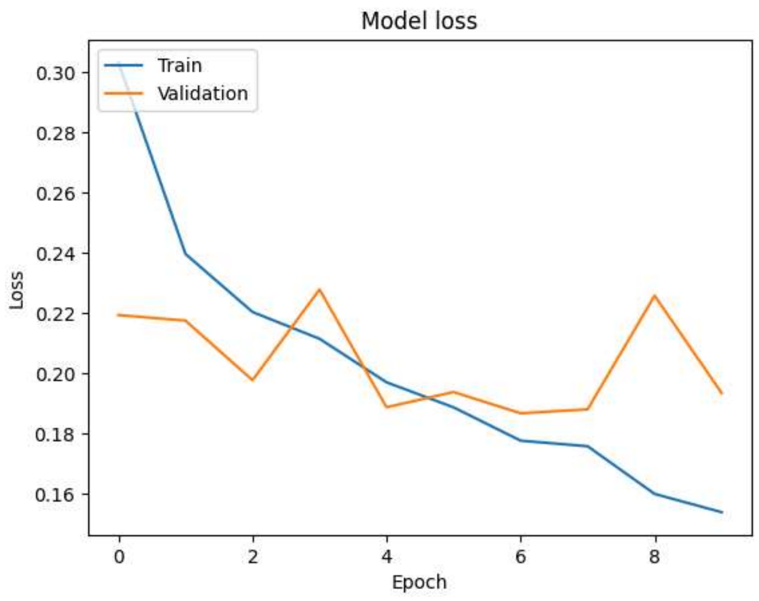
```
# Evaluate the model on the test data
loss, accuracy = model.evaluate(validation_generator)
print('Test accuracy:', accuracy)
```

```
157/157 ━━━━━━━━━━━━━━━━━━━━ 15s 94ms/step - accuracy: 0.9075 - loss: 0.2201
Test accuracy: 0.9172000288963318
```

```
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()
```


Model loss
Train
Validation
Loss
Epoch
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0
2
4
6
8