```
import os

from tensorflow.keras import layers
from tensorflow.keras import Model
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_ker
    -O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

from tensorflow.keras.applications.inception_v3 import InceptionV3

local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

pre_trained_model = InceptionV3(input_shape = (150, 150, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

for layer in pre_trained_model.layers:
  layer.trainable = False

# pre_trained_model.summary()

last_layer = pre_trained_model.get_layer('mixed7')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output
```

```
--2020-09-17 07:23:57--  https://storage.googleapis.com/mledu-datasets/inception_v3_weig
Resolving storage.googleapis.com (storage.googleapis.com)... 64.233.166.128, 74.125.71.1
Connecting to storage.googleapis.com (storage.googleapis.com)|64.233.166.128|:443... cor
HTTP request sent, awaiting response... 200 OK
Length: 87910968 (84M) [application/x-hdf]
Saving to: ‘/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5’

/tmp/inception_v3_w 100%[===================>]  83.84M   259MB/s    in 0.3s

2020-09-17 07:23:58 (259 MB/s) - ‘/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_n

last layer output shape:  (None, 7, 7, 768)
```

```
from tensorflow.keras.optimizers import RMSprop

# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
```

```
# Add a fully connected layer with 1,024 hidden units and ReLU activation
x = layers.Dense(1024, activation='relu')(x)
# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)
# Add a final sigmoid layer for classification
x = layers.Dense  (1, activation='sigmoid')(x)

model = Model( pre_trained_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001),
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])



!wget --no-check-certificate \
        https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip \
       -O /tmp/cats_and_dogs_filtered.zip

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import zipfile

local_zip = '//tmp/cats_and_dogs_filtered.zip'

zip_ref = zipfile.ZipFile(local_zip, 'r')

zip_ref.extractall('/tmp')
zip_ref.close()

# Define our example directories and files
base_dir = '/tmp/cats_and_dogs_filtered'

train_dir = os.path.join( base_dir, 'train')
validation_dir = os.path.join( base_dir, 'validation')


train_cats_dir = os.path.join(train_dir, 'cats') # Directory with our training cat pictures
train_dogs_dir = os.path.join(train_dir, 'dogs') # Directory with our training dog pictures
validation_cats_dir = os.path.join(validation_dir, 'cats') # Directory with our validation ca
validation_dogs_dir = os.path.join(validation_dir, 'dogs')# Directory with our validation dog

train_cat_fnames = os.listdir(train_cats_dir)
train_dog_fnames = os.listdir(train_dogs_dir)

# Add our data-augmentation parameters to ImageDataGenerator
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)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator( rescale = 1.0/255. )

# Flow training images in batches of 20 using train_datagen generator
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size = 20,
                                                    class_mode = 'binary',
                                                    target_size = (150, 150))

# Flow validation images in batches of 20 using test_datagen generator
validation_generator =  test_datagen.flow_from_directory( validation_dir,
                                                          batch_size  = 20,
                                                          class_mode  = 'binary',
                                                          target_size = (150, 150))
```

```
--2020-09-17 07:24:41--  https://storage.googleapis.com/mledu-datasets/cats_and_dogs_fil
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.71.128, 74.125.133.1
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.71.128|:443... conn
HTTP request sent, awaiting response... 200 OK
Length: 68606236 (65M) [application/zip]
Saving to: ‘/tmp/cats_and_dogs_filtered.zip’

/tmp/cats_and_dogs_ 100%[===================>]  65.43M   118MB/s    in 0.6s

2020-09-17 07:24:42 (118 MB/s) - ‘/tmp/cats_and_dogs_filtered.zip’ saved [68606236/68606

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

```
history = model.fit(
            train_generator,
            validation_data = validation_generator,
            steps_per_epoch = 100,
            epochs = 20,
            validation_steps = 50,
            verbose = 2)
```

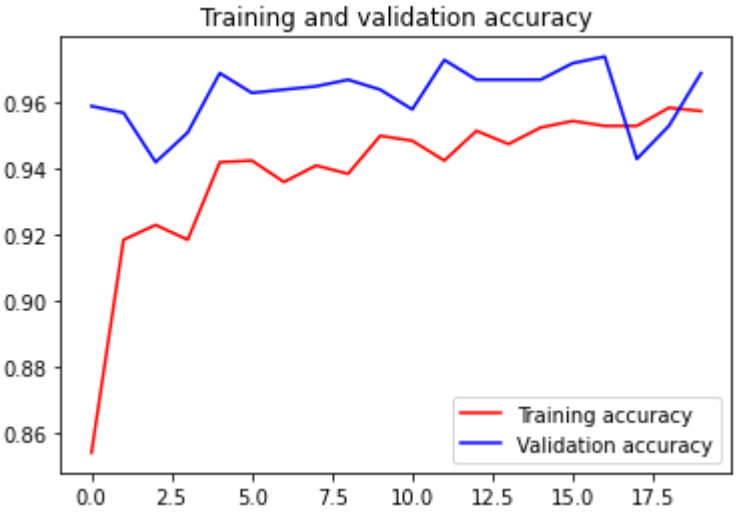
```
Epoch 1/20
100/100 - 23s - loss: 0.3641 - accuracy: 0.8540 - val_loss: 0.1126 - val_accuracy: 0.959
Epoch 2/20
100/100 - 22s - loss: 0.1915 - accuracy: 0.9185 - val_loss: 0.1189 - val_accuracy: 0.957
Epoch 3/20
100/100 - 22s - loss: 0.2165 - accuracy: 0.9230 - val_loss: 0.1699 - val_accuracy: 0.942
Epoch 4/20
100/100 - 22s - loss: 0.1969 - accuracy: 0.9185 - val_loss: 0.1474 - val_accuracy: 0.951
Epoch 5/20
100/100 - 22s - loss: 0.1734 - accuracy: 0.9420 - val_loss: 0.0971 - val_accuracy: 0.969
Epoch 6/20
100/100 - 22s - loss: 0.1813 - accuracy: 0.9425 - val_loss: 0.1219 - val_accuracy: 0.963
Epoch 7/20
100/100 - 22s - loss: 0.1840 - accuracy: 0.9360 - val_loss: 0.1279 - val_accuracy: 0.964
Epoch 8/20
100/100 - 21s - loss: 0.1691 - accuracy: 0.9410 - val_loss: 0.1265 - val_accuracy: 0.965
Epoch 9/20
100/100 - 22s - loss: 0.1630 - accuracy: 0.9385 - val_loss: 0.1139 - val_accuracy: 0.967
Epoch 10/20
100/100 - 22s - loss: 0.1463 - accuracy: 0.9500 - val_loss: 0.1269 - val_accuracy: 0.964
Epoch 11/20
100/100 - 22s - loss: 0.1425 - accuracy: 0.9485 - val_loss: 0.1456 - val_accuracy: 0.958
Epoch 12/20
100/100 - 22s - loss: 0.1462 - accuracy: 0.9425 - val_loss: 0.1060 - val_accuracy: 0.973
Epoch 13/20
100/100 - 22s - loss: 0.1423 - accuracy: 0.9515 - val_loss: 0.1197 - val_accuracy: 0.967
Epoch 14/20
```

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()


plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```