Open in Colab

```
# Import all the necessary files!
import os
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import Model


# Download the inception v3 weights
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5 \
    -O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

# Import the inception model
from tensorflow.keras.applications.inception_v3 import InceptionV3

# Create an instance of the inception model from the local pre-trained weights
local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

pre_trained_model = InceptionV3(input_shape = (150, 150, 3), ### YOUR CODE HERE
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

# Make all the layers in the pre-trained model non-trainable
for layer in pre_trained_model.layers:
  layer.trainable = False ### YOUR CODE HERE

# Print the model summary
pre_trained_model.summary()
```

```
__________________________________________________________________________________________________
batch_normalization_184 (BatchN (None, 3, 3, 384)    1152        conv2d_184[0][0]
__________________________________________________________________________________________________
activation_180 (Activation)     (None, 3, 3, 384)    0           batch_normalization_180[0][0]
__________________________________________________________________________________________________
activation_184 (Activation)     (None, 3, 3, 384)    0           batch_normalization_184[0][0]
__________________________________________________________________________________________________
conv2d_181 (Conv2D)             (None, 3, 3, 384)    442368      activation_180[0][0]
__________________________________________________________________________________________________
```

```
conv2d_182 (Conv2D)             (None, 3, 3, 384)    442368      activation_180[0][0]
__________________________________________________________________________________________________
conv2d_185 (Conv2D)             (None, 3, 3, 384)    442368      activation_184[0][0]
__________________________________________________________________________________________________
conv2d_186 (Conv2D)             (None, 3, 3, 384)    442368      activation_184[0][0]
__________________________________________________________________________________________________
average_pooling2d_17 (AveragePo (None, 3, 3, 2048)   0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_179 (Conv2D)             (None, 3, 3, 320)    655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_181 (BatchN (None, 3, 3, 384)    1152        conv2d_181[0][0]
__________________________________________________________________________________________________
batch_normalization_182 (BatchN (None, 3, 3, 384)    1152        conv2d_182[0][0]
__________________________________________________________________________________________________
batch_normalization_185 (BatchN (None, 3, 3, 384)    1152        conv2d_185[0][0]
__________________________________________________________________________________________________
batch_normalization_186 (BatchN (None, 3, 3, 384)    1152        conv2d_186[0][0]
__________________________________________________________________________________________________
conv2d_187 (Conv2D)             (None, 3, 3, 192)    393216      average_pooling2d_17[0][0]
__________________________________________________________________________________________________
batch_normalization_179 (BatchN (None, 3, 3, 320)    960         conv2d_179[0][0]
__________________________________________________________________________________________________
activation_181 (Activation)     (None, 3, 3, 384)    0           batch_normalization_181[0][0]
__________________________________________________________________________________________________
activation_182 (Activation)     (None, 3, 3, 384)    0           batch_normalization_182[0][0]
__________________________________________________________________________________________________
activation_185 (Activation)     (None, 3, 3, 384)    0           batch_normalization_185[0][0]
__________________________________________________________________________________________________
activation_186 (Activation)     (None, 3, 3, 384)    0           batch_normalization_186[0][0]
__________________________________________________________________________________________________
batch_normalization_187 (BatchN (None, 3, 3, 192)    576         conv2d_187[0][0]
__________________________________________________________________________________________________
activation_179 (Activation)     (None, 3, 3, 320)    0           batch_normalization_179[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, 3, 3, 768)    0           activation_181[0][0]
                                                                 activation_182[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 3, 3, 768)    0           activation_185[0][0]
                                                                 activation_186[0][0]
__________________________________________________________________________________________________
activation_187 (Activation)     (None, 3, 3, 192)    0           batch_normalization_187[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 3, 3, 2048)   0           activation_179[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_3[0][0]
                                                                 activation_187[0][0]
==================================================================================================
Total params: 21,802,784
```

```
Trainable params: 0
Non-trainable params: 21,802,784
__________________________________________________________________________________________________
```

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

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

```
# Define a Callback class that stops training once accuracy reaches 99.9%
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('accuracy')>0.999):
      print("\nReached 99.9% accuracy so cancelling training!")
      self.model.stop_training = True
```

```
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)### YOUR CODE HERE)(x)
# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)### YOUR CODE HERE)(x)
# Add a final sigmoid layer for classification
x = layers.Dense  (1, activation='sigmoid')(x)### YOUR CODE HERE)(x)

model = Model( pre_trained_model.input, x) ### YOUR CODE HERE, x)

model.compile(optimizer = RMSprop(learning_rate=0.0001),
              loss = 'binary_crossentropy', ### YOUR CODE HERE,
              metrics = ['accuracy'] ### YOUR CODE HERE,
              )

model.summary()
```

```
__________________________________________________________________________________________________
activation_155 (Activation)     (None, 7, 7, 192)    0           batch_normalization_155[0][0]
```

activation_160 (Activation)	(None, 7, 7, 192)	0	batch_normalization_160[0][0]
conv2d_156 (Conv2D)	(None, 7, 7, 192)	258048	activation_155[0][0]
conv2d_161 (Conv2D)	(None, 7, 7, 192)	258048	activation_160[0][0]
batch_normalization_156 (BatchN	(None, 7, 7, 192)	576	conv2d_156[0][0]
batch_normalization_161 (BatchN	(None, 7, 7, 192)	576	conv2d_161[0][0]
activation_156 (Activation)	(None, 7, 7, 192)	0	batch_normalization_156[0][0]
activation_161 (Activation)	(None, 7, 7, 192)	0	batch_normalization_161[0][0]
average_pooling2d_15 (AveragePo	(None, 7, 7, 768)	0	mixed6[0][0]
conv2d_154 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_157 (Conv2D)	(None, 7, 7, 192)	258048	activation_156[0][0]
conv2d_162 (Conv2D)	(None, 7, 7, 192)	258048	activation_161[0][0]
conv2d_163 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_15[0][0]
batch_normalization_154 (BatchN	(None, 7, 7, 192)	576	conv2d_154[0][0]
batch_normalization_157 (BatchN	(None, 7, 7, 192)	576	conv2d_157[0][0]
batch_normalization_162 (BatchN	(None, 7, 7, 192)	576	conv2d_162[0][0]
batch_normalization_163 (BatchN	(None, 7, 7, 192)	576	conv2d_163[0][0]
activation_154 (Activation)	(None, 7, 7, 192)	0	batch_normalization_154[0][0]
activation_157 (Activation)	(None, 7, 7, 192)	0	batch_normalization_157[0][0]
activation_162 (Activation)	(None, 7, 7, 192)	0	batch_normalization_162[0][0]
activation_163 (Activation)	(None, 7, 7, 192)	0	batch_normalization_163[0][0]
mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_154[0][0] activation_157[0][0] activation_162[0][0] activation_163[0][0]
flatten (Flatten)	(None, 37632)	0	mixed7[0][0]

```
dense (Dense)                (None, 1024)              38536192    flatten[0][0]
_____________________________________________________________________________________________
dropout (Dropout)            (None, 1024)              0           dense[0][0]
_____________________________________________________________________________________________
dense_1 (Dense)              (None, 1)                 1025        dropout[0][0]
=============================================================================================
Total params: 47,512,481
Trainable params: 38,537,217
Non-trainable params: 8,975,264
_____________________________________________________________________________________________
```

```
# Get the Horse or Human dataset
!gdown --id 1onaG42NZft3wCE1WH0GDEbUhu75fedP5

# Get the Horse or Human Validation dataset
!gdown --id 1LYeusSEIiZQpwN-mthh5nKdA75VsKG1U

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import zipfile

test_local_zip = './horse-or-human.zip'
zip_ref = zipfile.ZipFile(test_local_zip, 'r')
zip_ref.extractall('./training')

val_local_zip = './validation-horse-or-human.zip'
zip_ref = zipfile.ZipFile(val_local_zip, 'r')
zip_ref.extractall('./validation')

zip_ref.close()
```

```
Downloading...
From: https://drive.google.com/uc?id=1onaG42NZft3wCE1WH0GDEbUhu75fedP5
To: /content/horse-or-human.zip
100% 150M/150M [00:01<00:00, 97.2MB/s]
Downloading...
From: https://drive.google.com/uc?id=1LYeusSEIiZQpwN-mthh5nKdA75VsKG1U
To: /content/validation-horse-or-human.zip
100% 11.5M/11.5M [00:00<00:00, 101MB/s]
```

```
# Define our example directories and files
train_dir = './training'
```

```
validation_dir = './validation'

# Directory with our training horse pictures
train_horses_dir = os.path.join(train_dir, 'horses') ### YOUR CODE HERE
# Directory with our training humans pictures
train_humans_dir = os.path.join(train_dir, 'humans') ### YOUR CODE HERE
# Directory with our validation horse pictures
validation_horses_dir = os.path.join(validation_dir, 'horses') ### YOUR CODE HERE
# Directory with our validation humanas pictures
validation_humans_dir = os.path.join(validation_dir, 'humans') ### YOUR CODE HERE

train_horses_fnames = os.listdir(train_horses_dir) ### YOUR CODE HERE)
train_humans_fnames = os.listdir(train_humans_dir) ### YOUR CODE HERE)
validation_horses_fnames = os.listdir(validation_horses_dir) ### YOUR CODE HERE)
validation_humans_fnames = os.listdir(validation_humans_dir) ### YOUR CODE HERE)

print(len(train_horses_fnames)) ### YOUR CODE HERE)
print(len(train_humans_fnames)) ### YOUR CODE HERE)
print(len(validation_horses_fnames)) ### YOUR CODE HERE)
print(len(validation_humans_fnames)) ### YOUR CODE HERE)
```

```
500
527
128
128
```

```
# Add our data-augmentation parameters to ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255., ### YOUR CODE HERE)
                                   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. ) ### YOUR CODE HERE)

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

```
Found 1027 images belonging to 2 classes.
Found 256 images belonging to 2 classes.
```

```
# Run this and see how many epochs it should take before the callback
# fires, and stops training at 99.9% accuracy
# (It should take less than 100 epochs)
callbacks = myCallback() ### YOUR CODE HERE)
history = model.fit( ### YOUR CODE HERE)
            train_generator,
            validation_data = validation_generator,
            steps_per_epoch = 100,
            epochs = 100,
            validation_steps = 5,
            verbose = 2,
            callbacks=callbacks)
```

```
Epoch 1/100
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batche
100/100 - 13s - loss: 0.0242 - accuracy: 0.9903 - val_loss: 0.0169 - val_accuracy: 0.9900
```

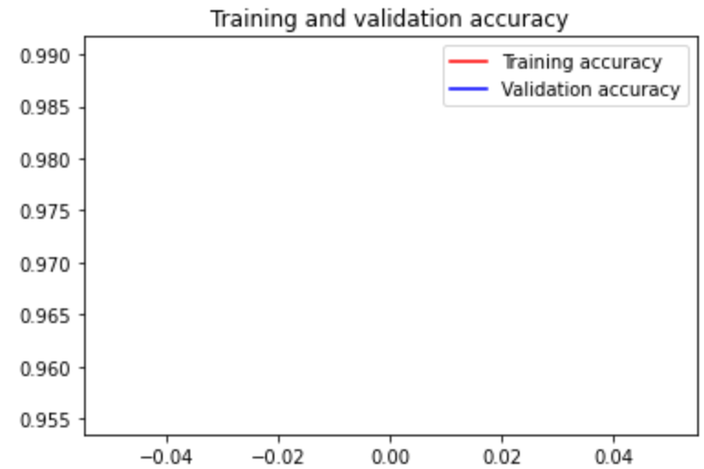
```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

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>
```

✓ 13s completed at 4:33 PM