Assignment 3

Dataset link
https://github.com/ThinamXx/Horse.vs.Human_Classification/blob/master/Dataset/validation-horse-or-human.zip

```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os
import zipfile
import tensorflow as tf

from google.colab import files
from keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip \
    -O /tmp/horse-or-human.zip
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/validation-horse-or-human.zip \
    -O /tmp/validation-horse-or-human.zip
local_zip = "/tmp/horse-or-human.zip"
zip_ref = zipfile.ZipFile(local_zip, "r")
zip_ref.extractall("/tmp/horse-or-human")
zip_ref.close()
local_zip = "/tmp/validation-horse-or-human.zip"
zip_ref = zipfile.ZipFile(local_zip, "r")
zip_ref.extractall("/tmp/validation-horse-or-human")
zip_ref.close()
# Directory with our training horse pictures
train_horse_dir = os.path.join("/tmp/horse-or-human/horses")

# Directory with our training human pictures
train_human_dir = os.path.join("/tmp/horse-or-human/humans")

# Directory with our validation horse pictures
validation_horse_dir = os.path.join("/tmp/validation-horse-or-human/horses")

# Directory with our validation human pictures
validation_human_dir = os.path.join("/tmp/validation-horse-or-human/humans")
# Training horse directory
train_horse_names = os.listdir(train_horse_dir)
print(train_horse_names[:10])

# Training human directory
train_human_names = os.listdir(train_human_dir)
print(train_human_names[:10])
```

```
# Validation horse directory
validation_horse_names = os.listdir(validation_horse_dir)
print(validation_horse_names[:10])

# Validation human directory
validation_human_names = os.listdir(validation_human_dir)
print(validation_human_names[:10])
# Training horses
print(f"Total training horse images: {len(os.listdir(train_horse_dir))}")

# Training humans
print(f"Total training humans images: {len(os.listdir(train_human_dir))}")

# Validation horses
print(f"Total validation horse images:
{len(os.listdir(validation_horse_dir))}")

# Validation humans
print(f"Total validation humans images:
{len(os.listdir(validation_human_dir))}")
# Parameters for our graph
nrows = 4
ncols = 4

# Index for iterating over images
pic_index = 0

# Setup matplotlib figure
fig = plt.gcf()
fig.set_size_inches(ncols*4, nrows*4)

pic_index += 8
next_horse_px = [os.path.join(train_horse_dir, fname) for fname in
train_horse_names[pic_index-8:pic_index]]
next_human_px = [os.path.join(train_human_dir, fname) for fname in
train_human_names[pic_index-8:pic_index]]

for i, img_path in enumerate(next_horse_px+next_human_px):
  # Set subplots
  sp = plt.subplot(nrows, ncols, i+1)
  sp.axis("Off")

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
# Building Convolutional Neural Network from scratch
model = tf.keras.models.Sequential([
                                    # The first convolution
                                    # Input image has 3 bytes color
                                    tf.keras.layers.Conv2D(16, (3, 3),
activation="relu", input_shape=(300, 300, 3)),
                                    tf.keras.layers.MaxPooling2D(2, 2),
                                    # The second convolution
                                    tf.keras.layers.Conv2D(32, (3, 3),
activation="relu"),
                                    tf.keras.layers.MaxPooling2D(2, 2),
```

```
                                    # The third convolution
                                    tf.keras.layers.Conv2D(64, (3, 3),
activation="relu"),
                                    tf.keras.layers.MaxPooling2D(2, 2),
                                    # The fourth convolution
                                    tf.keras.layers.Conv2D(64, (3, 3),
activation="relu"),
                                    tf.keras.layers.MaxPooling2D(2, 2),
                                    # The fifth convolution
                                    tf.keras.layers.Conv2D(64, (3, 3),
activation="relu"),
                                    tf.keras.layers.MaxPooling2D(2, 2),
                                    # Flatten the results to feed in Deep
Neural Network
                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(512,
activation="relu"),
                                    tf.keras.layers.Dense(1,
activation="sigmoid")
])
model.summary()
# Compile the Model
model.compile(loss="binary_crossentropy",
              optimizer=RMSprop(lr=0.001),
              metrics=["accuracy"])
train_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)

# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
    "/tmp/horse-or-human",
    target_size=(300, 300),
    batch_size=128,
    class_mode="binary"
)

# Flow validation images in batches of 32 using validation_datagen
generator
validation_generator = validation_datagen.flow_from_directory(
    "/tmp/validation-horse-or-human",
    target_size=(300, 300),
    batch_size=32,
    class_mode="binary"
)
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get("accuracy") > 0.99):
      print("\nReached 99% accurcy so stopping the execution of the
program!")
      self.model.stop_training = True

# Instantiation
callbacks = myCallback()
history = model.fit(
    train_generator,
    steps_per_epoch=8,
    epochs=15,
```

```
    verbose=1,
    validation_data=validation_generator,
    validation_steps=8
)
uploaded = files.upload()

for fn in uploaded.keys():
  # Predicting Images
  path = "/content/" + fn
  img = image.load_img(path, target_size=(300, 300))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  print(classes[0])
  if classes[0] > 0.5:
    print(fn, "is a human")
  else:
    print(fn, "is a horse")
```