Artificial Intelligence Term Project (Horses or Humans)

Dr. Aziz Al-otaibi

Stuednt name :

Abdulmajeed Mohammed Al-amri - 43803419

Meshal Bandar Hadhrawi – 43807307

Dataset information:

- Horses or Humans is a dataset of 300×300 images, created by Laurence Moroney, that is licensed CC-By-2.0 for anybody to use in learning or testing computer vision algorithms.

- The set contains 500 rendered images of various species of horse in various poses in various locations. It also contains 527 rendered images of humans in various poses and locations

- number of traning images = 1027
- image size 300x300 width x height.
- Number of classes 2 labels.

Load Dataset :

- Loading the dataset from google URL (zip file).
- Using OS library give us accses to the zipfile to unzip the file or data.

```
[43] !wget --no-check-certificate \
        https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip \
        -O /tmp/horse-or-human.zip
```

```
--2020-12-04 23:15:04--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.7.176, 142.250.73.208, 142.250.73.240, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.7.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 149574867 (143M) [application/zip]
Saving to: '/tmp/horse-or-human.zip'

/tmp/horse-or-human 100%[===================>] 142.65M   295MB/s    in 0.5s

2020-12-04 23:15:05 (295 MB/s) - '/tmp/horse-or-human.zip' saved [149574867/149574867]
```

```
[13] import os
     import zipfile

     local_zip = '/tmp/horse-or-human.zip'
     zip_ref = zipfile.ZipFile(local_zip, 'r')
     zip_ref.extractall('/tmp/horse-or-human')
     zip_ref.close()
```

Display a batch of 8 horses and 8 humans pictures:

```
[28] %matplotlib inline

     import matplotlib.pyplot as plt
     import matplotlib.image as mpimg

     # Parameters for our graph; we'll output images in a 4x4 configuration
     nrows = 4
     ncols = 4

     # Index for iterating over images
     pic_index = 0
```

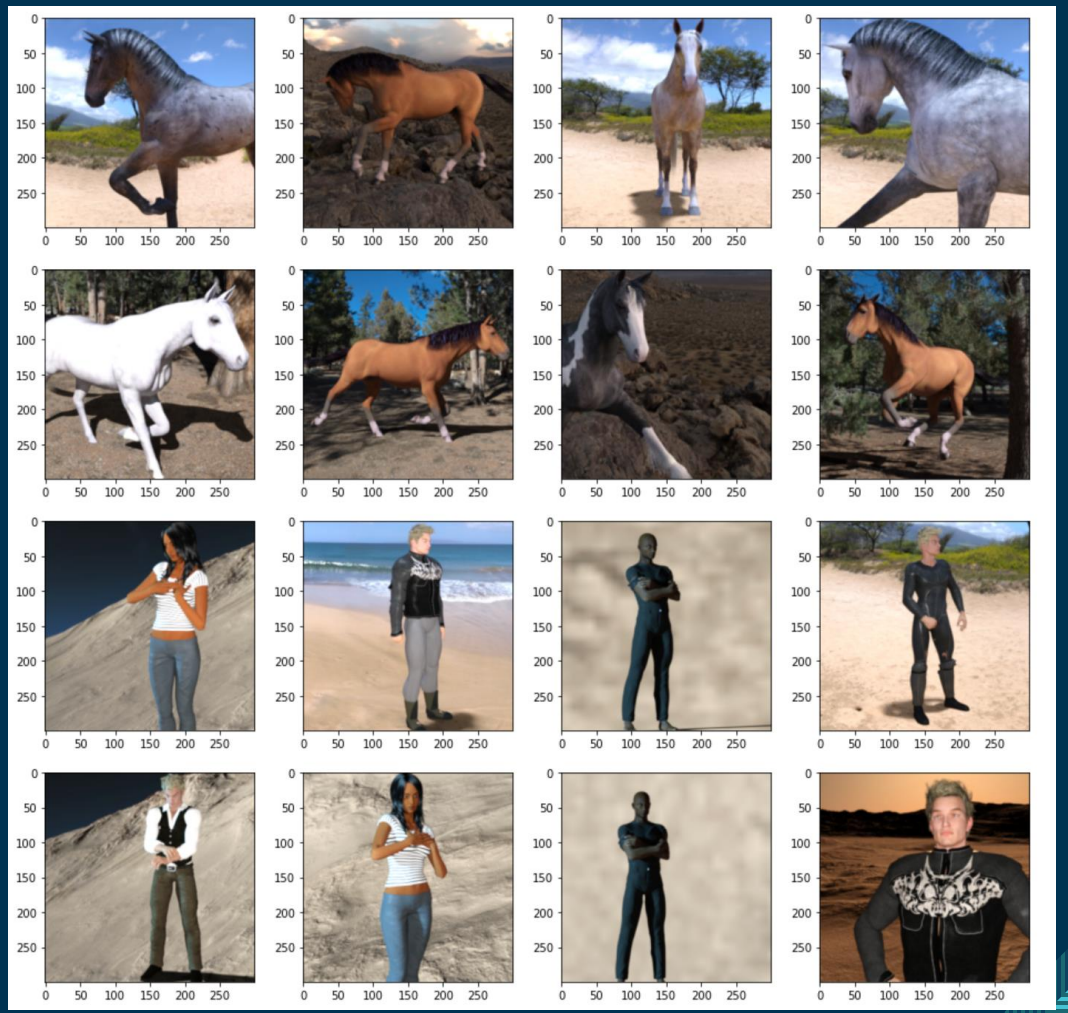
```
     # Set up matplotlib fig, and size it to fit 4x4 pics
     fig = plt.gcf()
     fig.set_size_inches(ncols * 4, nrows * 4)

     pic_index += 8
     next_horse_pix = [os.path.join(train_horse_dir, fname)
                       for fname in train_horse_names[pic_index-8:pic_index]]
     next_human_pix = [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_pix+next_human_pix):
       # Set up subplot; subplot indices start at 1
       sp = plt.subplot(nrows, ncols, i + 1)
       # Don't show axes (or gridlines)
       sp.axis('on')

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

     plt.show()
```



Model :

- **Building a Small Model from Scratch :**

  1.import tensorflow.

  2.add convolutional layers

```
import tensorflow as tf
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 300x300 with 3 bytes color
    # This is the first convolution
    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 into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1 where 0 for 1 class ('horses') and 1 for the other ('humans')
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

- **The model.summary() method call prints a summary of the NN**

```
[ ]  model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1606144
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,704,097
Trainable params: 1,704,097
Non-trainable params: 0
```

Training(Epoch,Batch_size,Activation_function)

Note:

- Batch_size : 128
- Activation_function : ReLU
- ReLU stands for rectified linear unit, and is a type of activation function.

- We have 1027 images in the dataset and 2 classes

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_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/',  # This is the source directory for training images
        target_size=(300, 300),
        batch_size=128,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')
```

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

Training(Epoch,Batch_size,Activation_function)

Note:

- Steps_per_Epoch
  = (1027 / 128) = 8

- verbose = 1, which includes both progress bar and one line per epoch

```
history = model.fit(
      train_generator,
      steps_per_epoch=8,
      epochs=15,
      verbose=1)
```

```
Epoch 1/15
8/8 [==============================] - 5s 677ms/step - loss: 0.7410 - accuracy: 0.5829
Epoch 2/15
8/8 [==============================] - 6s 764ms/step - loss: 0.7678 - accuracy: 0.6630
Epoch 3/15
8/8 [==============================] - 5s 674ms/step - loss: 0.3936 - accuracy: 0.8610
Epoch 4/15
8/8 [==============================] - 5s 671ms/step - loss: 0.2927 - accuracy: 0.8710
Epoch 5/15
8/8 [==============================] - 6s 747ms/step - loss: 0.3728 - accuracy: 0.8574
Epoch 6/15
8/8 [==============================] - 6s 753ms/step - loss: 0.1532 - accuracy: 0.9492
Epoch 7/15
8/8 [==============================] - 5s 664ms/step - loss: 0.1652 - accuracy: 0.9388
Epoch 8/15
8/8 [==============================] - 5s 670ms/step - loss: 0.0538 - accuracy: 0.9833
Epoch 9/15
8/8 [==============================] - 5s 663ms/step - loss: 1.7270 - accuracy: 0.8287
Epoch 10/15
8/8 [==============================] - 5s 665ms/step - loss: 0.2149 - accuracy: 0.9210
Epoch 11/15
8/8 [==============================] - 5s 659ms/step - loss: 0.0695 - accuracy: 0.9711
Epoch 12/15
8/8 [==============================] - 6s 754ms/step - loss: 0.0241 - accuracy: 0.9944
Epoch 13/15
8/8 [==============================] - 5s 652ms/step - loss: 0.0287 - accuracy: 0.9933
Epoch 14/15
8/8 [==============================] - 5s 668ms/step - loss: 0.0245 - accuracy: 0.9889
Epoch 15/15
8/8 [==============================] - 5s 674ms/step - loss: 0.0272 - accuracy: 0.9889
```

Result:

- You can click on Choose File, after selecting the file the test will be performed and give the result

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

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")
```

Result:

After selecting an image from the horses file

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

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")
```

```
Choose Files horse01-0.png
• horse01-0.png(image/png) - 151700 bytes, last modified: 10/22/2019 - 100% done
Saving horse01-0.png to horse01-0.png
[0.]
horse01-0.png is a horse
```

After selecting an image from the human file

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

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")
```

```
Choose Files human01-00.png
• human01-00.png(image/png) - 160725 bytes, last modified: 10/22/2019 - 100% done
Saving human01-00.png to human01-00.png
[1.]
human01-00.png is a human
```

Conclusion

- Finally In this project, we selected a dataset (Human_Horses) that is in Tenserflow, and built a neural network that uses more than one layer, and train the model and achieving high accuracy with minimum lost , and give the data for the training model and predicted the results.

3 Dec 2020