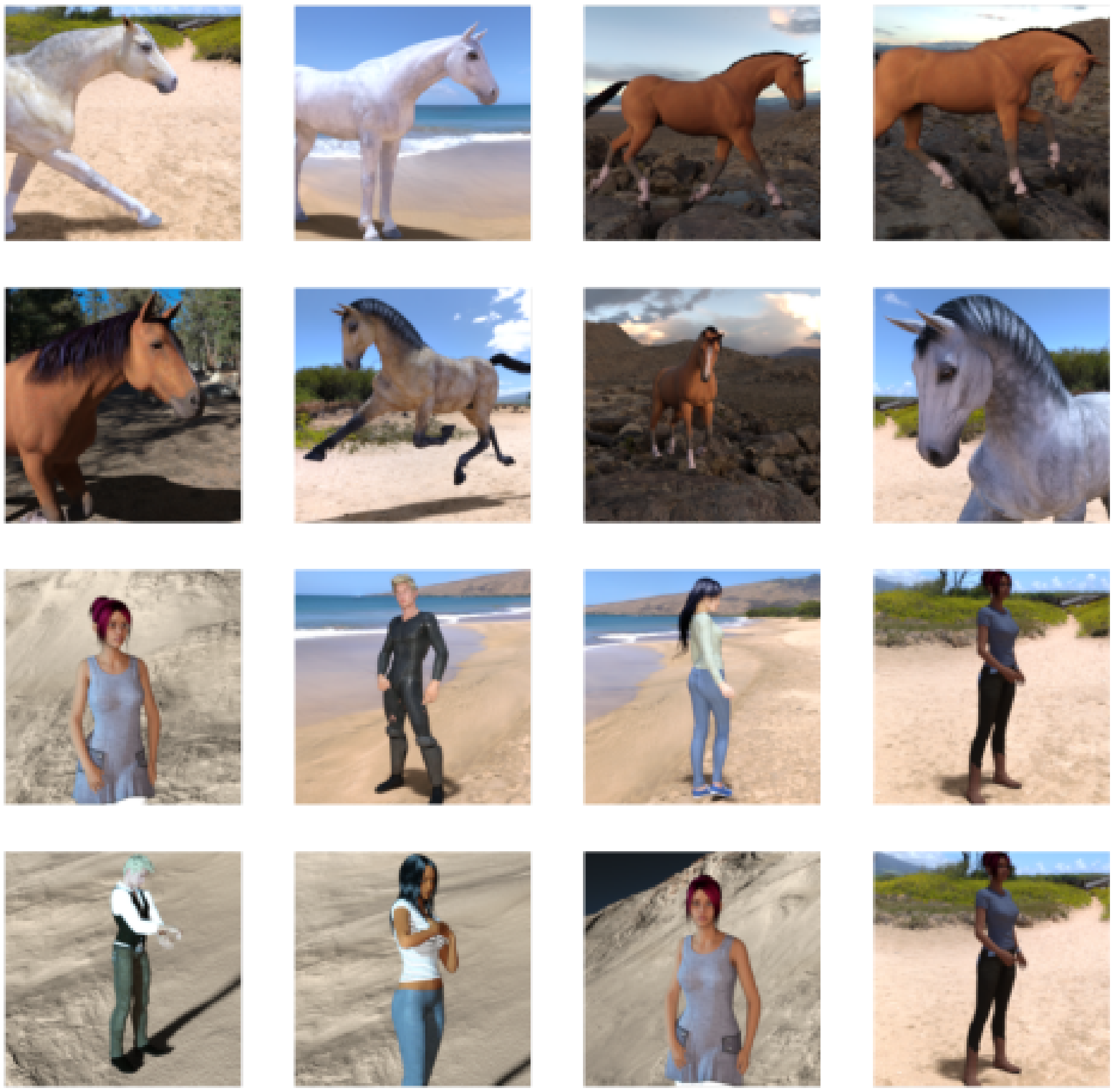
Human-Horse Image Classification

Problem Statement :
To classify images as horse or human using image processing in Machine Learning.

Introduction:
This project is using the tensorflow, keras, numpy and matplotlib libraries.
The model is trained using horse or human dataset.
The model is a sequential convolutional neural network model, with one input and one output layer along with five hidden layers.
CNN is a supervised machine learning algorithm.
It is used for image classification and computer vision.
The model will take input of a 300x300 image and classify it accordingly into horse or human.
The training accuracy of this model is 0.9883.
Validation accuracy of it is 0.8711.
The output prints the images uploaded along with their respective predicted classes.

Dataset Information:
Link for Training and validation dataset :
https://www.kaggle.com/datasets/sanikamal/horses-or-humans-dataset

Training: examples
For horses and for humans, 8 images per dataset are displayed below.



Code :

```
# horse or human
# training dataset
!wget
https://storage.googleapis.com/tensorflow-1-public/course2/week3/horse-or
human.zip

# validation dataset
!wget
https://storage.googleapis.com/tensorflow-1-public/course2/week3/validatio
n-horse-or-human.zip
```

```
# Unzip the dataset
import zipfile

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

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

zip_ref.close()
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'),
    tf.keras.layers.Dropout(0.5),
    # 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')
])
model.summary()
# compiling the model
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['accuracy'])



# generating images of the same size
from tensorflow.keras.preprocessing.image import ImageDataGenerator



train_datagen = ImageDataGenerator(rescale=1/255)
validation_datagen = ImageDataGenerator(rescale=1/255)



train_generator = train_datagen.flow_from_directory(
                './horse-or-human/',
        target_size=(300, 300),
        batch_size=128,
        class_mode='binary')

validation_generator = validation_datagen.flow_from_directory(
                './validation-horse-or-human/',
        target_size=(300, 300),
        batch_size=32,
        class_mode='binary')
# training the model
history = model.fit(
      train_generator,
        steps_per_epoch=8,
        epochs=14,
        verbose=1,
        validation_data = validation_generator,
        validation_steps=8)

# saving it to os
model_json = model.to_json()
with open("model.json","w") as json_file:
  json_file.write(model_json)
  model.save_weights("m.h5")
# uploading images and predicting its class
%pylab inline
import numpy as np
from google.colab import files
from keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg


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 /= 255
  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:
    img = mpimg.imread(path)
    imgplot = plt.imshow(img)
    plt.show()

    print(fn + " is a human")
    print()
  else:
    img = mpimg.imread(path)
    imgplot = plt.imshow(img)
    plt.show()

    print(fn + " is a horse")
    print()
```

Conclusion:

The model trained classified the given four images with 0% error. Convolutional NeuralNetwork is the basic architecture for image processing.

Convolutional neural networks (CNNs) have accomplished astonishing achievements across a variety of domains, including medical research, and an increasing interest has emerged in radiology. Although deep learning has become a dominant method in a variety of complex tasks such as image classification and object detection, it is not a panacea. Being familiar with key concepts and advantages of CNN as well as limitations of deep learning is essential in order to leverage it in radiology research with the goal of improving radiologist performance and, eventually, patient care.

References:

1. https://www.coursera.org/learn/introduction-tensorflow/supplement/XFWSt/get-hands-on-with-computer-vision-lab-1
2. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
3. https://www.tensorflow.org/
4. https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/
5. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9