Assignment 3

Create a model to perform binary classification between horse and human images using convolutional neural

Dataset available in Tensorflow datasets

In [229…
```
#  import all necessary libraries
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Activation, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as k
from keras.utils import load_img
```

In [230…
```
#  will set the image size
img_height, img_width = 100, 100
```

In [231…
```
#  lets read the train and test data
train_data= 'horse-or-human'
test_data= 'validation-horse-or-human'
```

In [232…
```
train_data, test_data
```

Out[232…
```
('horse-or-human', 'validation-horse-or-human')
```

In [233…
```
image= load_img('horse-or-human/horses/horse01-7.png')
image
```

Out[233…

In [234…
```
image1= load_img('horse-or-human/humans/human01-15.png')
image1
```

Out[234…

In [235…
```
"""
This part is to check the data format i.e the RGB channel is coming first or last so
whatever it may be, the model will check first and then input shape will be fed acco
"""
if k.image_data_format() == 'channels_first':
    input_shape = (3, img_height, img_width)
else:
    input_shape = (img_height, img_width, 3)
```

In [236…
```
model = Sequential()
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))
```

In [237…
```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [238…
```
train_samples = 1027
test_samples = 256
```

```
epochs = 2
batch_size = 16
```

In [239…

```
train_datagen = ImageDataGenerator()

test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(
    train_data,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    test_data,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

model.fit_generator(
    train_generator,
    steps_per_epoch= train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps= test_samples // batch_size)
```

```
Found 1027 images belonging to 2 classes.
Found 256 images belonging to 2 classes.
Epoch 1/2
C:\Users\Manjula\AppData\Local\Temp/ipykernel_14648/348857211.py:17: UserWarning: `M
odel.fit_generator` is deprecated and will be removed in a future version. Please us
e `Model.fit`, which supports generators.
  model.fit_generator(
64/64 [==============================] - 10s 151ms/step - loss: 4.6179 - accuracy:
0.7695 - val_loss: 1.7395 - val_accuracy: 0.8008
Epoch 2/2
64/64 [==============================] - 9s 137ms/step - loss: 0.2197 - accuracy: 0.
9159 - val_loss: 1.8242 - val_accuracy: 0.7500
```

Out[239…

```
<keras.callbacks.History at 0x1f6f3b31d30>
```

In [240…

```
# lets test one of the image from test folder
# image= load_img("validation-horse-or-human/horses/horse3-440.png")

image= cv2.imread('validation-horse-or-human/horses/horse3-440.png')
```

In [241…

```
# need to resize the image as its trained in the same shape
image1= np.resize(image,(1,100,100,3))

image1
```

Out[241…

```
array([[[[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         ...,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        [[255, 255, 255],
```

```
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[ 50,  41,  38],
        [ 50,  41,  38],
        [ 52,  42,  38],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [ 38,  32,  34],
        [ 47,  39,  36],
        [ 48,  40,  36]]]], dtype=uint8)
```

In [242…

```
# predicting

label= model.predict(image1)
label
```

```
1/1 [==============================] - 0s 87ms/step
```

Out[242…

```
array([[0.20724708]], dtype=float32)
```

In [243…

```
if label[0][0] > 0.5:
    print("Human")
else:
    print("Horse")
```

```
Horse
```

In [244…

```
# another test from validation folder

image2= load_img('validation-horse-or-human/humans/valhuman01-20.png')
image3= np.resize(image2,(1,100,100,3))
```

```
label1= model.predict(image3)
print(label1)
```

```
1/1 [==============================] - 0s 20ms/step
[[0.6581151]]
```

In [245…

```
if label1[0][0] > 0.5:
    print("Human")
else:
    print("Horse")
```

```
Human
```

In [256…

```
# test performed on images taken from web; human/horse

img1= load_img('human_image.png')
img2= np.resize(img1,(1,100,100,3))
class1= model.predict(img2)
print(class1)
```

```
1/1 [==============================] - 0s 16ms/step
[[0.53259194]]
```

In [257…

```
if class1[0][0] > 0.5:
    print("Our model predicts the given image as Human")
else:
    print("Our model predicts the given image as Horse")
```

```
Our model predicts the given image as Human
```

In [258…

```
# test performed on images taken from web; human/horse

img3= load_img('horse_image.png')
img4= np.resize(img3,(1,100,100,3))
class2= model.predict(img4)
print(class2)

if class2[0][0] > 0.5:
    print("Our model predicts the given image as Human")
else:
    print("Our model predicts the given image as Horse")
```

```
1/1 [==============================] - 0s 16ms/step
[[0.01526007]]
Our model predicts the given image as Horse
```

In [255…

```
# img2     #resized image in RGB channels
```

Out[255…

```
array([[[[229, 225, 213],
         [227, 223, 211],
         [224, 220, 208],
         ...,
         [227, 223, 211],
         [229, 225, 213],
         [229, 225, 213]],

        [[227, 223, 211],
         [225, 221, 209],
         [225, 221, 209],
         ...,
```

```
       [223, 219, 208],
       [223, 219, 208],
       [224, 220, 209]],

      [[226, 222, 211],
       [226, 222, 211],
       [227, 223, 212],
       ...,
       [227, 223, 212],
       [226, 222, 213],
       [226, 222, 213]],

      ...,

      [[188, 181, 165],
       [189, 182, 166],
       [189, 182, 164],
       ...,
       [167, 159, 146],
       [165, 157, 144],
       [166, 158, 145]],

      [[164, 156, 143],
       [166, 158, 145],
       [169, 161, 148],
       ...,
       [210, 204, 190],
       [211, 205, 191],
       [211, 205, 191]],

      [[212, 206, 192],
       [212, 206, 192],
       [212, 206, 192],
       ...,
       [219, 213, 199],
       [219, 213, 197],
       [219, 213, 197]]], dtype=uint8)
```

In [ ]:

In [ ]: