CNN image classification - Messi Vs Ronaldo

This project is basically about getting any images from the web for classification using the following steps

1. Loading image data
2. Cleaning image data
3. Scaling/normalization/preprocessing of image data
4. Model training
5. Testing model

Loading and cleaning image data

In [39]:
```
# Importing packages
import tensorflow as tf
import cv2
import imghdr
import os
import matplotlib.pyplot as plt
```

In [90]:
```
# Looping and cleaning :- Messi Vs Ronaldo images
data_dir = 'MvR' # name of image folder on my computer
img_ex = ['jpeg', 'jpg', 'bmp', 'png'] # type of image extensions to consider
for img_class in os.listdir(data_dir):
    for img in os.listdir(os.path.join(data_dir, img_class)):
        img_path = os.path.join(data_dir, img_class, img)
        try:
            img = cv2.imread(img_path)
            tip = imghdr.what(img_path)
            if tip not in img_ex:
                print(f'Not here {img_path}')
                os.remove(img_path)
        except Exception as e:
            print(f'img issues {img_path}')
```

In [114]:
```
# Labeling, suffling and storing data into memory using tf.keras.utils

import numpy as np
import matplotlib.pyplot as plt

data = tf.keras.utils.image_dataset_from_directory('MvR')# Total of 120 images with 2 classes
data_itera = data.as_numpy_iterator()
batch = data_itera.next() # each batch contains 32 images
batch[0].shape # 32 images of 256x256
batch[1] # label of each images
```

Found 120 files belonging to 2 classes.

Out[114]:
```
array([0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1])
```

In [138]:
```
# Plotting the first four images with it's label

fig, ax = plt.subplots(ncols = 4)
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(img.astype(int)) # 0 = messi, 1 = ronaldo
    ax[idx].title.set_text(batch[1][idx])
```

Preprocessing

In [118]:
```
#Scaling the data with min - 0 and max - 1

data = tf.keras.utils.image_dataset_from_directory('MvR')
data = data.map(lambda x, y: (x/255, y))
```

Found 120 files belonging to 2 classes.

In [123]:
```
#Iterating over scaled data
scaled_itera = data.as_numpy_iterator()
scaled_batch = scaled_itera.next()
scaled_batch[0].max()
```

Out[123]:
```
1.0
```

In [124]:
```
# Plotting imaging of scaled data

fig, ax = plt.subplots(ncols = 4)
for idx, img in enumerate(scaled_batch[0][:4]):
    ax[idx].imshow(img) # 0 = messi, 1 = ronaldo
    ax[idx].title.set_text(scaled_batch[1][idx])
```

Traning data for analysis

In [129]:
```
#splitting data into training, validation and test data, all must be equall to the lenght of the data
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)+1
test_size = int(len(data)*.1)+1
print('Data_size: {len(data)}, train_size: {train_size}, val_size: {val_size}, test_size: {test_size}')
```

```
Data_size: 4, train_size: 2, val_size: 1, test_size: 1
```

In [73]:
```
# Take and skip the appropriate assingned ratio from each batch of data

train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

In [64]:
```
#Import and initiate modules needed for classification

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
model = Sequential()
```

In [66]:
```
#Addding layers of network
model.add(Conv2D(16, (3,3), 1, activation = 'relu', input_shape = (256,256,3)))
model.add(MaxPooling2D())

model.add(Conv2D(32, (3,3), 1, activation = 'relu'))
model.add(MaxPooling2D())

model.add(Conv2D(16, (3,3), 1, activation = 'relu'))
model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(256, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

In [67]:
```
# Compile my model
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

In [68]:
```
#Summary of model generated
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2  (None, 127, 127, 16)      0
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 16)        0
 g2D)

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 256)               3686656

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 3696625 (14.10 MB)
Trainable params: 3696625 (14.10 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [69]:
```
# This is call back/ track of model
logdir='logs'
tensorboard = tf.keras.callbacks.TensorBoard(log_dir = logdir)
```

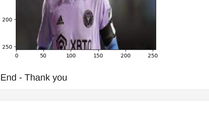
In [75]:
```
# 20 iterations
hist = model.fit(train, epochs=20, validation_data=val, callbacks = [tensorboard])
```

```
Epoch 1/20
2/2 [==============================] - 12s 5s/step - loss: 0.6795 - accuracy: 0.5000 - val_loss: 0.7470 - val_accuracy: 0.6250
Epoch 2/20
2/2 [==============================] - 8s 4s/step - loss: 1.1828 - accuracy: 0.4844 - val_loss: 0.7000 - val_accuracy: 0.5312
Epoch 3/20
2/2 [==============================] - 8s 5s/step - loss: 0.5666 - accuracy: 0.6719 - val_loss: 0.7038 - val_accuracy: 0.5938
Epoch 4/20
2/2 [==============================] - 6s 4s/step - loss: 0.6193 - accuracy: 0.6562 - val_loss: 0.7138 - val_accuracy: 0.5625
Epoch 5/20
2/2 [==============================] - 6s 3s/step - loss: 0.6033 - accuracy: 0.6406 - val_loss: 0.5868 - val_accuracy: 0.9375
Epoch 6/20
2/2 [==============================] - 8s 4s/step - loss: 0.5420 - accuracy: 0.8750 - val_loss: 0.5878 - val_accuracy: 0.8125
Epoch 7/20
2/2 [==============================] - 9s 5s/step - loss: 0.5132 - accuracy: 0.8438 - val_loss: 0.4925 - val_accuracy: 0.8438
Epoch 8/20
2/2 [==============================] - 6s 3s/step - loss: 0.4866 - accuracy: 0.8281 - val_loss: 0.4817 - val_accuracy: 0.7188
Epoch 9/20
2/2 [==============================] - 5s 3s/step - loss: 0.4907 - accuracy: 0.6875 - val_loss: 0.4361 - val_accuracy: 0.8125
Epoch 10/20
2/2 [==============================] - 5s 3s/step - loss: 0.4191 - accuracy: 0.8750 - val_loss: 0.3706 - val_accuracy: 0.9375
Epoch 11/20
2/2 [==============================] - 5s 3s/step - loss: 0.3596 - accuracy: 0.8750 - val_loss: 0.3033 - val_accuracy: 0.9062
Epoch 12/20
2/2 [==============================] - 5s 3s/step - loss: 0.3565 - accuracy: 0.8438 - val_loss: 0.3131 - val_accuracy: 0.8125
Epoch 13/20
2/2 [==============================] - 5s 3s/step - loss: 0.3101 - accuracy: 0.8594 - val_loss: 0.2114 - val_accuracy: 1.0000
Epoch 14/20
2/2 [==============================] - 5s 3s/step - loss: 0.2737 - accuracy: 0.9219 - val_loss: 0.2179 - val_accuracy: 0.9688
Epoch 15/20
2/2 [==============================] - 5s 3s/step - loss: 0.2253 - accuracy: 0.9531 - val_loss: 0.1553 - val_accuracy: 1.0000
Epoch 16/20
2/2 [==============================] - 5s 3s/step - loss: 0.1718 - accuracy: 0.9844 - val_loss: 0.1454 - val_accuracy: 0.9688
Epoch 17/20
2/2 [==============================] - 5s 3s/step - loss: 0.1307 - accuracy: 0.9375 - val_loss: 0.0882 - val_accuracy: 1.0000
Epoch 18/20
2/2 [==============================] - 5s 3s/step - loss: 0.0954 - accuracy: 1.0000 - val_loss: 0.0977 - val_accuracy: 1.0000
Epoch 19/20
2/2 [==============================] - 5s 3s/step - loss: 0.0676 - accuracy: 0.9844 - val_loss: 0.1283 - val_accuracy: 0.9688
Epoch 20/20
2/2 [==============================] - 5s 3s/step - loss: 0.0680 - accuracy: 0.9688 - val_loss: 0.0920 - val_accuracy: 0.9688
```

At Epoch 20, loss: 0.0680 , accuracy: 0.9688 , val_loss: 0.0920, val_accuracy: 0.9688.

In [131]:
```
# plotting training_loss, val_loss
plt.plot(hist.history['loss'], color = 'black', label = 'training_loss')
plt.plot(hist.history['val_loss'], color = 'blue', label = 'val_loss')
plt.legend()
```

Out[131]:
```
<matplotlib.legend.Legend at 0x2543546d680>
```



In [132]:
```
# plotting training_accuracy, val_accuracy
plt.plot(hist.history['accuracy'], color = 'black', label = 'training_accuracy')
plt.plot(hist.history['val_accuracy'], color = 'blue', label = 'val_accuracy')
plt.legend()
```

Out[132]:
```
<matplotlib.legend.Legend at 0x2543585955b>
```

In [79]:
```
#Parameters to evaluate test data
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

In [134]:
```
#Evaluate test data in batch

for batches in test.as_numpy_iterator():
    x, y = batch
    ypred = model.predict(x)
    pre.update_state(y, ypred)
    re.update_state(y, ypred)
    acc.update_state(y, ypred)

print(f'Precision: {pre.result().numpy()}, Recall: {re.result().numpy()}, Binaryaccuracy: {acc.result().numpy()}')
```

```
1/1 [==============================] - 0s 383ms/step
Precision: 1.0, Recall: 0.9821428656578064, Binaryaccuracy: 0.9895833134651184
```

In [135]:
```
#predicting new Ronaldo and Messi pictures
img = cv2.imread('cr.jpg')
size = tf.image.resize(img, (256,256))
plt.imshow(size.numpy().astype(int))# printing new image
```

Out[135]:
```
<matplotlib.image.AxesImage at 0x2543548a290>
```


In [137]:
```
# Correct prediction of Ronaldo
new_prod = model.predict(np.expand_dims(size/255, 0))
if new_prod < 0.5:
    print('GOAT, Messi')
else:
    print('Ronaldo')
print(new_prod)
```

```
1/1 [==============================] - 0s 60ms/step
Ronaldo
[[0.72990131]]
```

In [139]:
```
# classification of new messi image

img = cv2.imread('ms.jpg')
resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
new_prod = model.predict(np.expand_dims(resize/255, 0))
if new_prod < 0.5:
    print('GOAT, Messi')
else:
    print('Ronaldo')
print(new_prod)
```

```
1/1 [==============================] - 0s 60ms/step
GOAT, Messi
[[0.12249901]]
```


End - Thank you

In [ ]:
```
```