ARIES PROJECT

April 24, 2021

[13]:

```
import pandas as pd
import sys,os
import numpy as np
import keras
from keras.models import Sequential
from keras.layers.core import  Dropout
from keras.layers import Dense, Conv2D, MaxPooling2D , Flatten
from tensorflow.keras.layers.experimental import preprocessing
from keras.optimizers import SGD
import cv2
import np_utils
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
```

[2]:

```
emotion_data = pd.read_csv('C:\\Users\\HP\\Desktop\\fer2013.csv')
```

[3]:

```
X_train = []
y_train = []
X_test = []
y_test = []
for index, row in emotion_data.iterrows():
    k = row['pixels'].split(" ")
    if row['Usage'] == 'Training':
        X_train.append(np.array(k))
        y_train.append(row['emotion'])
    elif row['Usage'] == 'PublicTest':
        X_test.append(np.array(k))
        y_test.append(row['emotion'])
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)
#print(X_train.shape)
#print(X_test.shape)
#print(y_test.shape)
#print(y_test.shape)
X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)
```

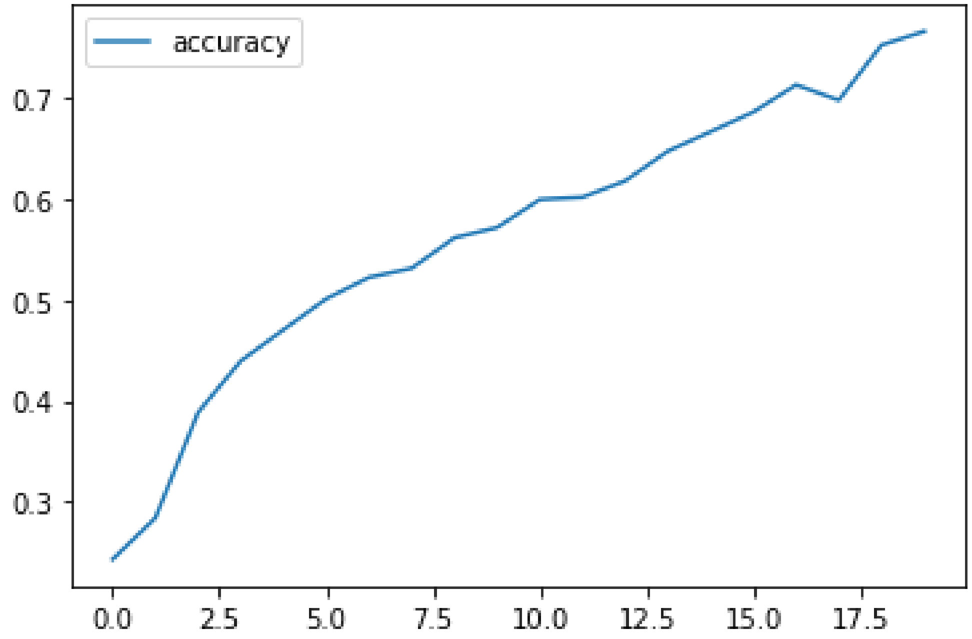
```
X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)
#print(X_train.shape)

y_train= keras.utils.to_categorical(y_train, num_classes=7)
y_test =keras.utils.to_categorical(y_test, num_classes=7)
```

[14]:

```
model = Sequential()

#1st layer
model.add(Conv2D(64,(5,5),activation='relu',input_shape=(48,48,1)))
preprocessing.RandomFlip(mode='horizontal')
preprocessing.RandomRotation(factor=0.05)
model.add(MaxPooling2D(pool_size=(5,5),strides=(2,2)))

#2nd layer
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#3rd layer
model.add(Conv2D(256,(3,3),activation='relu'))
model.add(Conv2D(256,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))


model.add(Flatten())

model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(7,activation='softmax'))
gen = ImageDataGenerator()
train_gen = gen.flow(X_train,y_train,batch_size=512)
gen1= ImageDataGenerator()
test_gen=gen1.flow(X_test,y_test,batch_size=512)
model.
 ↪compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit(train_gen,validation_data=test_gen,batch_size=512,epochs = 20)
history_frame = pd.DataFrame(history.history)
# history_frame.loc[:, ['loss', 'val_loss']].plot()
history_frame.loc[:, ['accuracy']].plot();
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer.h5")
```

```
Epoch 1/20
57/57 [==============================] - 333s 6s/step - loss: 3.2164 - accuracy:
0.2267 - val_loss: 1.8244 - val_accuracy: 0.2014
Epoch 2/20
57/57 [==============================] - 317s 6s/step - loss: 1.7918 - accuracy:
0.2594 - val_loss: 1.6372 - val_accuracy: 0.3625
Epoch 3/20
57/57 [==============================] - 307s 5s/step - loss: 1.5928 - accuracy:
0.3768 - val_loss: 1.5994 - val_accuracy: 0.3859
Epoch 4/20
57/57 [==============================] - 313s 5s/step - loss: 1.4850 - accuracy:
0.4268 - val_loss: 1.4373 - val_accuracy: 0.4544
Epoch 5/20
57/57 [==============================] - 325s 6s/step - loss: 1.4126 - accuracy:
0.4621 - val_loss: 1.3738 - val_accuracy: 0.4820
Epoch 6/20
57/57 [==============================] - 324s 6s/step - loss: 1.3196 - accuracy:
0.5014 - val_loss: 1.3005 - val_accuracy: 0.5093
Epoch 7/20
57/57 [==============================] - 323s 6s/step - loss: 1.2455 - accuracy:
0.5280 - val_loss: 1.3039 - val_accuracy: 0.4937
Epoch 8/20
57/57 [==============================] - 317s 6s/step - loss: 1.2677 - accuracy:
0.5156 - val_loss: 1.2738 - val_accuracy: 0.5177
Epoch 9/20
57/57 [==============================] - 292s 5s/step - loss: 1.1523 - accuracy:
0.5674 - val_loss: 1.3120 - val_accuracy: 0.5018
Epoch 10/20
57/57 [==============================] - 293s 5s/step - loss: 1.1477 - accuracy:
0.5685 - val_loss: 1.2689 - val_accuracy: 0.5185
Epoch 11/20
57/57 [==============================] - 292s 5s/step - loss: 1.0679 - accuracy:
0.5958 - val_loss: 1.2960 - val_accuracy: 0.5283
Epoch 12/20
57/57 [==============================] - 294s 5s/step - loss: 1.0363 - accuracy:
0.6144 - val_loss: 1.2462 - val_accuracy: 0.5358
Epoch 13/20
57/57 [==============================] - 307s 5s/step - loss: 0.9905 - accuracy:
0.6239 - val_loss: 1.2570 - val_accuracy: 0.5311
Epoch 14/20
57/57 [==============================] - 298s 5s/step - loss: 0.9414 - accuracy:
0.6512 - val_loss: 1.2798 - val_accuracy: 0.5489
Epoch 15/20
57/57 [==============================] - 294s 5s/step - loss: 0.8686 - accuracy:
0.6748 - val_loss: 1.2689 - val_accuracy: 0.5369
Epoch 16/20
57/57 [==============================] - 519s 9s/step - loss: 0.8699 - accuracy:
0.6773 - val_loss: 1.3989 - val_accuracy: 0.5294
```

```
Epoch 17/20
57/57 [==============================] - 485s 9s/step - loss: 0.7540 - accuracy:
0.7192 - val_loss: 1.3886 - val_accuracy: 0.5255
Epoch 18/20
57/57 [==============================] - 494s 9s/step - loss: 0.8511 - accuracy:
0.6844 - val_loss: 1.3640 - val_accuracy: 0.5561
Epoch 19/20
57/57 [==============================] - 520s 9s/step - loss: 0.6220 - accuracy:
0.7722 - val_loss: 1.4314 - val_accuracy: 0.5458
Epoch 20/20
57/57 [==============================] - 343s 6s/step - loss: 0.6123 - accuracy:
0.7739 - val_loss: 1.5063 - val_accuracy: 0.5656
```



[20]:

```
from keras.models import load_model
from time import sleep
#from keras.models import model_from_jason
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

#model = model_from_json(open("fer.json", "r").read())
#load weights
model.load_weights('fer.h5')
```

```
face_classifier = cv2.CascadeClassifier(r"C:
 ↪\Users\HP\Desktop\Emotion-recognition-master\haarcascade_files\haarcascade_frontalface_defaul
 ↪xml")
#classifier =load_model(r'C:
 ↪\Users\Admin\Desktop\PythonProject\EmotionDetectionCNN\model.h5')

emotion_labels = ['Angry','Disgust','Fear','Happy','Sad', 'Surprise', 'Neutral']

cap = cv2.VideoCapture(0)




while True:
    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)



        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi,axis=0)

            prediction =model.predict(roi)[0]
#             print(prediction)
            label=emotion_labels[prediction.argmax()]
            label_position = (x,y)
            cv2.putText(frame,label,label_position,cv2.
 ↪FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
        else:
            cv2.putText(frame,'No Faces',(30,80),cv2.
 ↪FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

[ ]:

```
```