DMBI

OEP Facial Expression Recognition

170420107005 Archil

170420107013 Vatsal

170420107014 Nevil

170420107022 Kartavya

170420107023 Dhruvin

Importing libraries for Data Processing

In [ ]:
```
import cv2
import pandas as pd
import glob
```

Converting Images to CSV

In [ ]:
```
disc={0:"angry",1:"disgust",2:"fear",3:"happy",4:"neutral",5:"sad",6:"s
urprise"}
m=int(input("Enter which data to process: "))
cv_img = []
```

```
s1=[]
for img in glob.glob("C:/Users/archi/OneDrive/Desktop/DMBI_OEP/images/"
+disc[m]+"/*.jpg"):
    n= cv2.imread(img)
    n=cv2.cvtColor(n, cv2.COLOR_BGR2GRAY)
    s=""
    for i in range(0,48):
        for j in range(0,48):
            s=s+' '+str(n[i][j])
    s1.append(s)
df=pd.DataFrame({"pixels":s1,"class":m})
writer = pd.ExcelWriter("C:/Users/archi/OneDrive/Desktop/DMBI_OEP/datas
et/"+disc[m]+".xlsx" )
df.to_excel(writer,'Sheet1',index=False)
writer.save()
```

In [ ]:

```
disc={0:"angry",1:"disgust",2:"fear",3:"happy",4:"neutral",5:"sad",6:"s
urprise"}
s1=[]
m1=[]
count=0
for m in range(7):
    print("Converting ",disc[m])
    for img in glob.glob("C:/Users/archi/OneDrive/Desktop/DMBI_OEP/imag
es/"+disc[m]+"/*.jpg"):
        #print(img)
        n= cv2.imread(img)
        n=cv2.cvtColor(n, cv2.COLOR_BGR2GRAY)
        s=""
        for i in range(0,48):
            for j in range(0,48):
                s=s+' '+str(n[i][j])
        s1.append(s)
        m1.append(m)
        count += 1
df=pd.DataFrame({"pixels":s1,"class":m1})
#writer = pd.ExcelWriter("H:/7th Sem/DBMI/5_OEP/dataset/training/train
2.csv")
df.to_csv("C:/Users/archi/OneDrive/Desktop/DMBI_OEP/dataset/train.csv",
index=False)
print("Total Images processed: ",count)
```

Importing Libraries for Training CNN Model.

In [ ]:
```
import sys, os
import pandas as pd
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormaliz
ation
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import ReduceLROnPlateau, TensorBoard,
EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model
```

In [ ]:
```
BASEPATH = './'
sys.path.insert(0, BASEPATH)
os.chdir(BASEPATH)
MODELPATH = './model.h5'

num_features = 64
num_labels = 7
batch_size = 64
epochs = 100
width, height = 48, 48

data = pd.read_csv('C:/Users/archi/OneDrive/Desktop/DMBI_OEP/dataset/fe
r2013.csv') #read dataset csv file
```

In [ ]:
```
# images stored in csv are resized into 2D array
pixels = data['pixels'].tolist() # 1
```

```
faces = []
for pixel_sequence in pixels:
    face = [int(pixel) for pixel in pixel_sequence.split(' ')] # 2
    face = np.asarray(face).reshape(width, height) # 3
    faces.append(face.astype('float32'))

faces = np.asarray(faces)
faces = np.expand_dims(faces, -1) # 6
```

In [ ]:
```
#One hot encoding of emotion array
emotions = pd.get_dummies(data['emotion']).as_matrix() # 7
```

In [ ]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(faces, emotions, te
st_size=0.1, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, tes
t_size=0.1, random_state=41)
```

In [ ]:
```
#initialize sequential model
model = Sequential()

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', i
nput_shape=(width, height, 1), data_format='channels_last', kernel_regu
larizer=l2(0.01)))
model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', p
adding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu',
 padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu',
 padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='rel
u', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='rel
u', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='re
lu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='re
lu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(2*2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*num_features, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(num_labels, activation='softmax'))
```

In [ ]:
```
#setting compiler parameters
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(lr=0.001),
              metrics=['accuracy'])
```

In [ ]:
```
lr_reducer = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience
=3, verbose=1)
```

```
tensorboard = TensorBoard(log_dir='./logs')
early_stopper = EarlyStopping(monitor='val_loss', min_delta=0, patience
=8, verbose=1, mode='auto')
checkpointer = ModelCheckpoint(MODELPATH, monitor='val_loss', verbose=1
, save_best_only=True)
```

In [ ]:
```
#training model
model.fit(np.array(X_train), np.array(y_train),
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(np.array(X_test), np.array(y_test)),
          shuffle=True,
          callbacks=[lr_reducer,early_stopper, checkpointer])
```

Using trained model to get output

In [1]:
```
from tensorflow.keras.models import load_model
import cv2
import numpy as np
```

In [2]:
```
emotion_dict = {0: "Angry", 1: "Disgust", 2: "Fear", 3: "Happy", 4: "Sa
d", 5: "Surprise", 6: "Neutral"}
model = load_model('model.h5',compile=False)

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    frame=cv2.flip(frame,1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_defau
lt.xml')
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
        roi_gray = gray[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray
, (48, 48)), -1), 0)
        cv2.normalize(cropped_img, cropped_img, alpha=0, beta=1, norm_t
ype=cv2.NORM_L2, dtype=cv2.CV_32F)
        prediction = model.predict(cropped_img)
        cv2.putText(frame, emotion_dict[int(np.argmax(prediction))], (x
, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1, cv2.LINE_AA)

    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

```
WARNING:tensorflow:From C:\Users\archi\Anaconda3\lib\site-packages\tens
orflow_core\python\ops\resource_variable_ops.py:1630: calling BaseResou
rceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

```
In [ ]:
```