Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Семинарска работа по предметот Дигитално процесирање на слика на тема:

„Препознавање на емоции во видео во реално време“

-Техничка документација-

Ментор:

Проф. Ивица Димитровски

Изработил:

Мартин Анѓелевски 216064

1. Вовед

Целта на оваа семинарска работа е креирање на апликација која овозможува препознавање на емоции. Во конкретниов случај се врши препознавање на емоции во видео од камерата на уредот на кој се извршува апликацијата во реално време. Од технички аспект, апликацијата се состои од два дела: претпроцесирање на податочно множество и тренирање на конволуциска невронска мрежа, и скрипта која го користи истренираниот модел за препознавање на емоции. Целосниот проект е достапен на GitHub [1].

2. Тренирање на конволуциска невронска мрежа

Датасетот врз кој се тренира моделот е fer2013.csv [2]. Тој се состои од 35887 слики класифицирани според емоција во 7 класи и тоа: anger, disgust, fear, happiness, sad, surprise, neutral. Во ‘facial-emotion-recognition-using-cnn.ipynb’ најпрво се прават потребните претпроцесирања. Се вчитуваат податоците во pandas dataframe и се енкодираат категориските променливи со техниката One Hot Encoding.

```
data = pd.read_csv("./fer2013.csv")
data.shape
```

```
(35887, 3)
```

```
labels = to_categorical(data[['emotion']], num_classes=7)
labels
```

```
array([[0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

Со цел тежините во моделот да немаат преголема варијација пикселите се скалираат користејќи StandardScaler. Потоа се дели множеството на тренирачко, тестирачко и валидациско.

3.3 Standardization

We will change the data structure to feed the standart scaler to implement standardization process to our data. Since StandartScaler() function only takes 2 dimensional array we will reshape the data then apply our scaler to make the mean zero and standart deviation as unit.

```
pixels = train_pixels.reshape((35887*2304,1))
```

```
scaler = StandardScaler()
pixels = scaler.fit_transform(pixels)
```

3.4 Reshaping the data (48,48)

After that, we will reshape the data to make our image pixels ready to split operation.

```
pixels = pixels.reshape((35887, 48, 48,1))
```

3.5 Train test validation split

Now, we have 35887 images with each containing 48x48 pixels. We will split the data into train,test and Validation data to feed and evaluate and validate our data with the ratio of 10%.

```
X_train, X_test, y_train, y_test = train_test_split(pixels, labels, test_size=0.1, shuffle=False)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, shuffle=False)
```

Се гради конволуциска невронска мрежа. За ова се користи библиотеката keras. Секој од слоевите користи активациска функција ‘relu’ освен последниот кој користи ‘softmax’ и со тоа дава предвидувања за седумте класи. Креирањето на моделот изгледа вака:

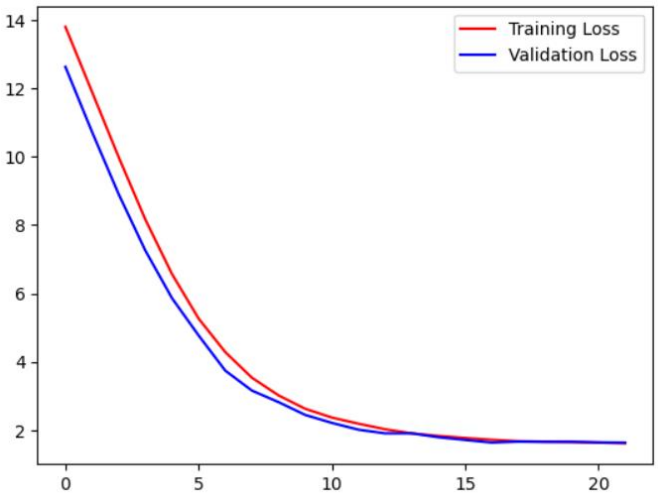
```
model= tf.keras.models.Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shape=(48, 48,1)))
model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
```

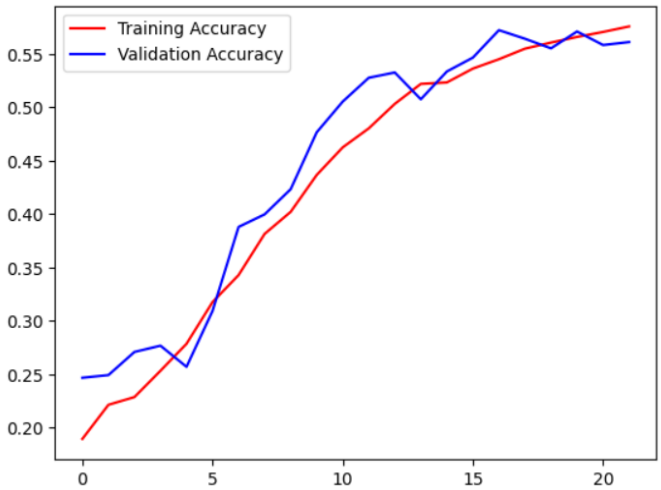
```
model.add(Dense(512,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))
```

По креирањето на моделот тој се тренира и валидира на соодветните делови од податочното множество. За тренирање на моделот во 50 епохи потребно беше време од 6 часа. Голем дел од овој notebook е преземен од [3] со тоа што направив некои промени кои сметав дека се потребни: скалирањето на пикселите, подесување на learning rate и испробување на различен број на епохи при тренирање. Резултатите од перформансот на моделот се следниве:

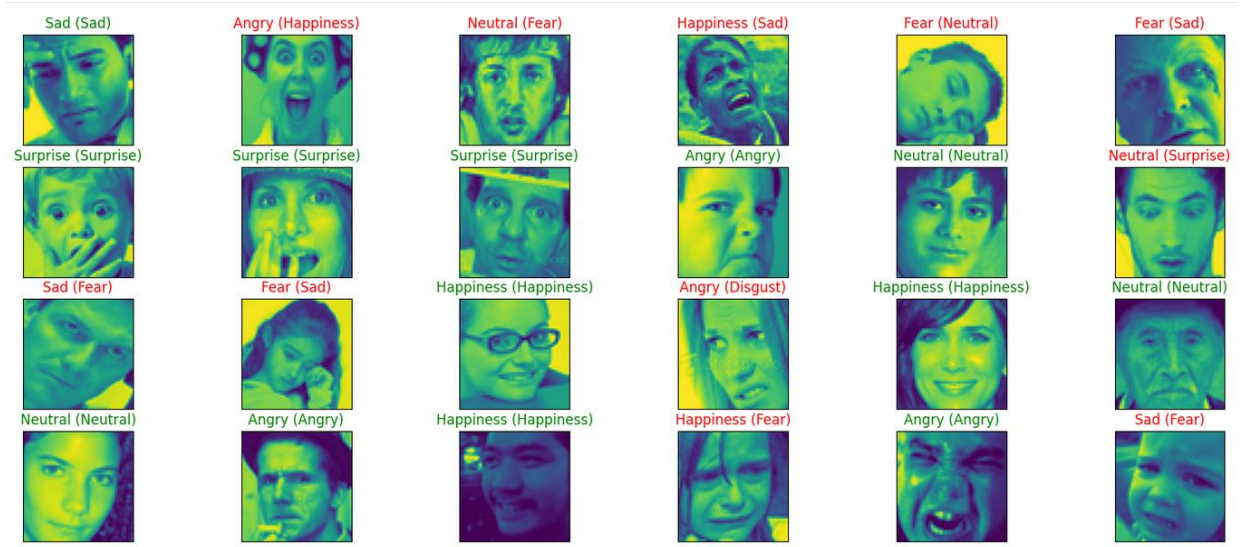
```
from sklearn.metrics import classification_report
print(classification_report(np.argmax(y_test, axis = 1 ),y_pred,digits=3))
```

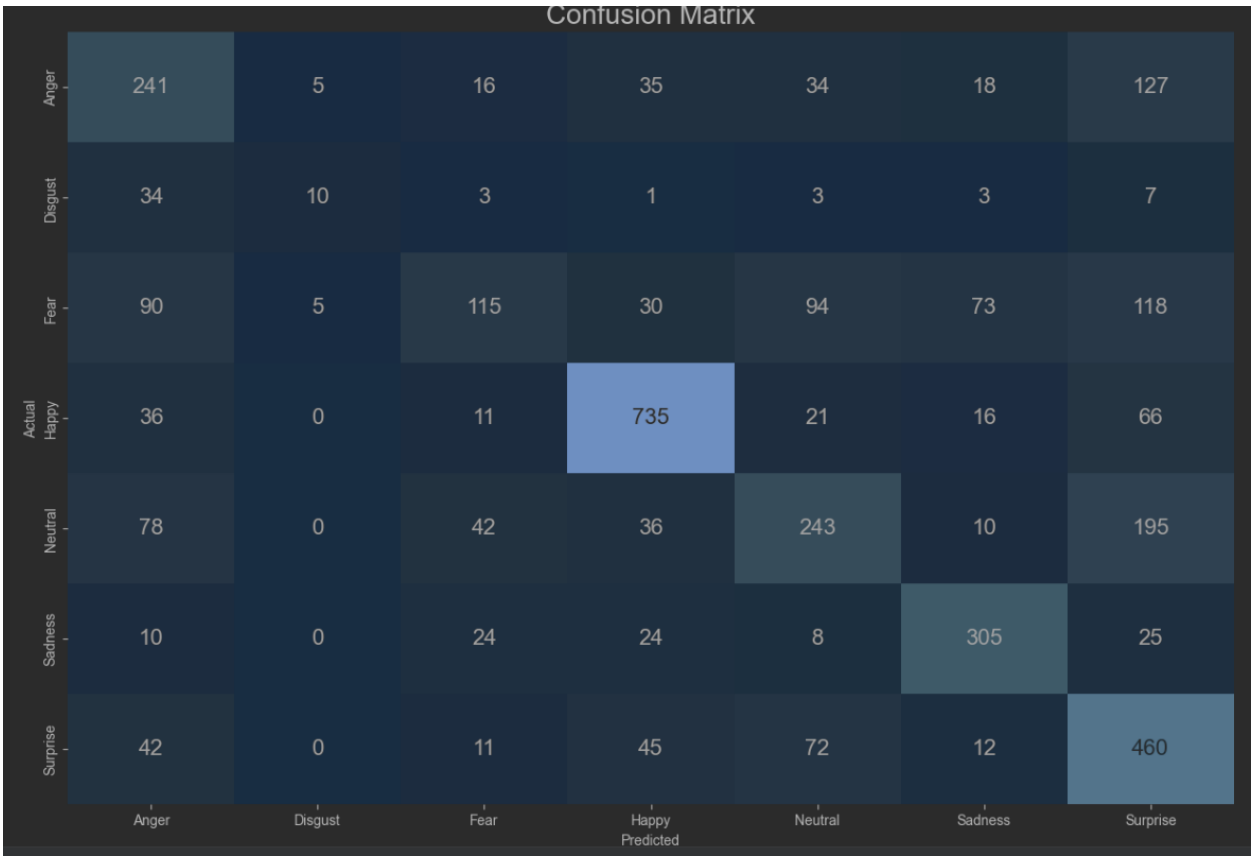
```
              precision    recall  f1-score   support

           0      0.454     0.506     0.479       476
           1      0.500     0.164     0.247        61
           2      0.518     0.219     0.308       525
           3      0.811     0.831     0.821       885
           4      0.512     0.402     0.450       604
           5      0.698     0.770     0.732       396
           6      0.461     0.717     0.561       642

    accuracy                          0.588      3589
   macro avg      0.565     0.516     0.514      3589
weighted avg      0.590     0.588     0.572      3589
```





На следната слика може да се забележи кои класи ги предвидува моделот за одредени слики од тестирачкото множество, а подолу гледаме и confusion matrix кој ни кажува како се класифицирани сите слики од тестирачкото множество и кои од нив се точно предвидени.





Моделот перформира со точност од 58.8% што може да биде подобрено, меѓутоа поради ограниченост од процесирачка моќ во делот за препознавање на емоција во реално време ќе го користам овој модел.

3. Демо апликација за препознавање на емоции

Скриптата за овој дел се наоѓа во main.py. Најпрво се вчитува истренираниот модел, како и скалерот кој беше користен за нормализација на пикселите. За издвојување на лицата од моментален frame како регион од интерес се користи готовиот Haar Cascade classifier за детекција на лица кој го нуди библиотеката OpenCV [4].

```
# Load the pre-trained model
model = load_model('best_model.h5')
scaler = load(open('scaler.pkl', 'rb'))

# Load the pre-trained Haar Cascade classifier for face detection
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Со cv2.VideoCapture(0) апликацијата добива пристап до интегрираната камера. Секој frame најпрво се конвертира од BGR во црно-бела слика и со користење на каскадниот класификатор се детектираат лицата на таа слика. Регион од интерес се детектираните лица па затоа за секое детектирано лице во одреден frame правиме resize во димензии 48x48 бидејќи таква димензионалност е потребна за влез во моделот. Се исцртува рамка околу секое детектирано лице и се нормализира регионот од интерес.

```
# Start capturing video from the camera
cap = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break

    # Convert the captured frame to grayscale
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the frame
    faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))

    # Process each detected face
    for (x, y, w, h) in faces:
```

```
        # Extract the region of interest (ROI) containing the face
        roi_gray = gray_frame[y:y+h, x:x+w]
        roi_gray = cv2.resize(roi_gray, (48, 48),
interpolation=cv2.INTER_AREA)

        # Put a rectangle around the face (ROI)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 1)

        # Normalize the image and reshape for model input
        roi = roi_gray.astype("float")
        # roi = roi.reshape(-1, 1)
        # roi = scaler.transform(roi)
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)
```

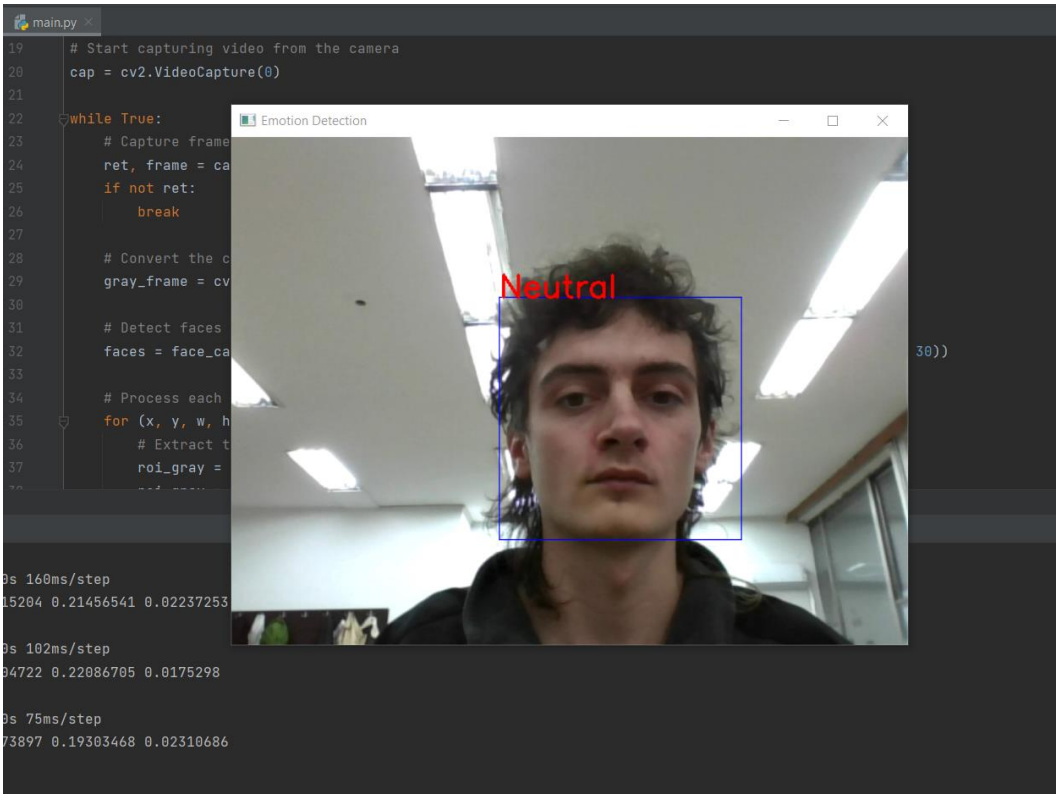
Секое лице (roi) се праќа на моделот за класификација при што моделот враќа предикции (веројатности по класа). Се зема класата за која моделот предвидува најголема веројатност и се испишува како текст на видеото.

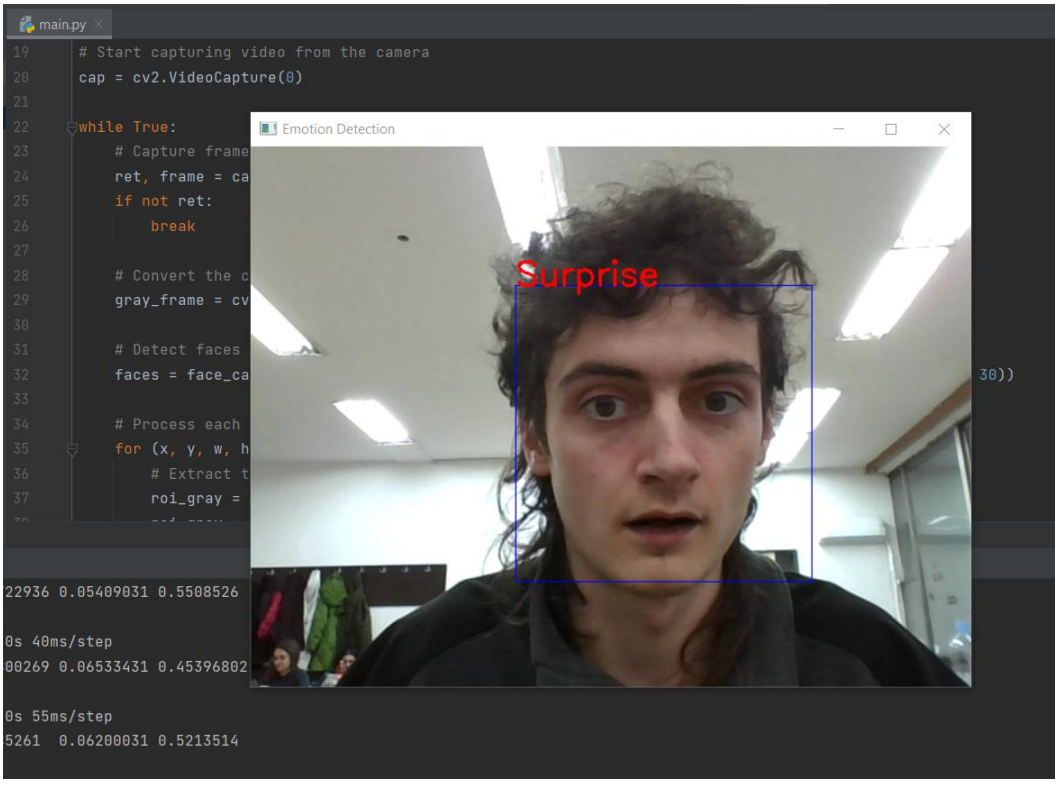
```
# Make a prediction
preds = model.predict(roi)[0]
print(preds)
emotion_label = emotion_dict[np.argmax(preds)]
# Display the emotion label on the frame
cv2.putText(frame, emotion_label, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 0, 255), 2)
```

Секој frame се прикажува во прозорец Emotion Detection се додека не се притисне ‘q’ на тастатура.

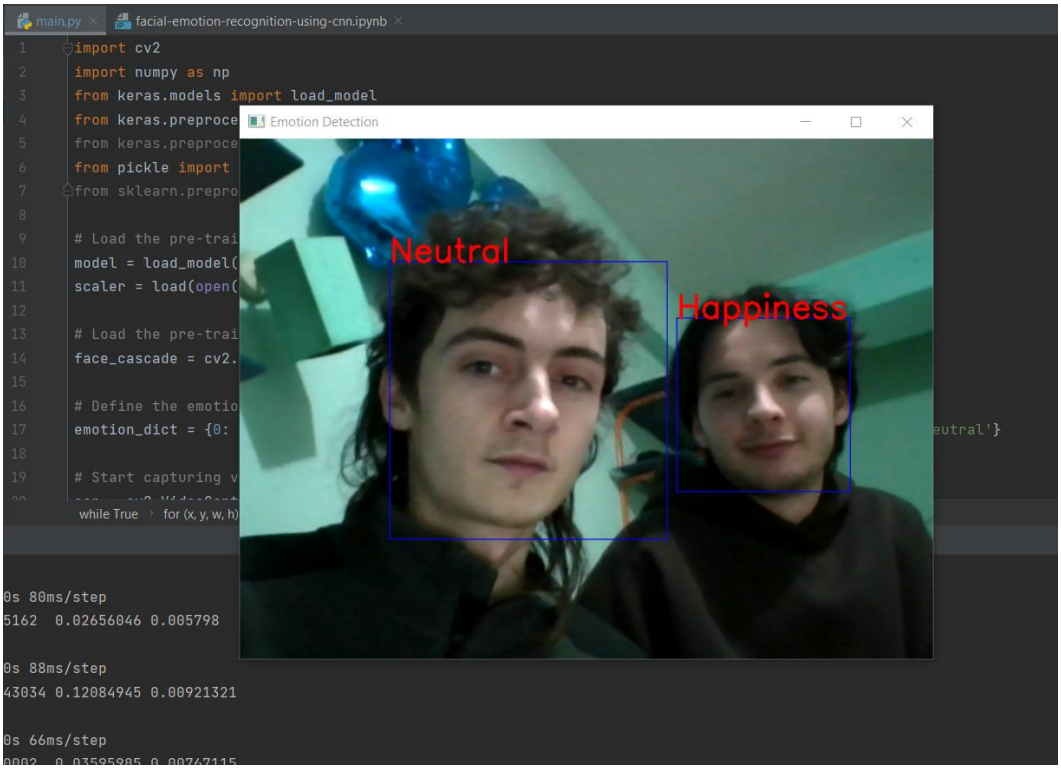
4. Користење на апликацијата

За да се стартува апликацијата се извршува main.py скриптата при што е потребно да се инсталирани библиотеките кои се користат. Се појавува прозорец Emotion Detection каде гледаме како работи препознавањето на емоции.





Апликацијата работи и со повеќе лица на камерата:


main.py
facial-emotion-recognition-using-cnn.ipynb
import cv2
import numpy as np
from keras.models import load_model
from keras.preproce
from keras.preproce
from pickle import
from sklearn.prepro
# Load the pre-trai
model = load_model(
scaler = load(open(
# Load the pre-trai
face_cascade = cv2.
# Define the emotio
emotion_dict = {0:
eutral'}
Start capturing v
while True
for (x, y, w, h)
Emotion Detection
Neutral
Happiness
0s 80ms/step
5162 0.02656046 0.005798
0s 88ms/step
43034 0.12084945 0.00921321
0s 66ms/step

5. Користена литература

[1] - https://github.com/angjelevski/EmotionDetection
[2] – https://www.kaggle.com/datasets/deadskull7/fer2013/
[3] – https://www.kaggle.com/code/oykuer/emotion-detection-using-cnn
[4] – https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
[5] - https://www.geeksforgeeks.org/emotion-detection-using-convolutional-neural-networks-cnns/