December 17

CNN

2023

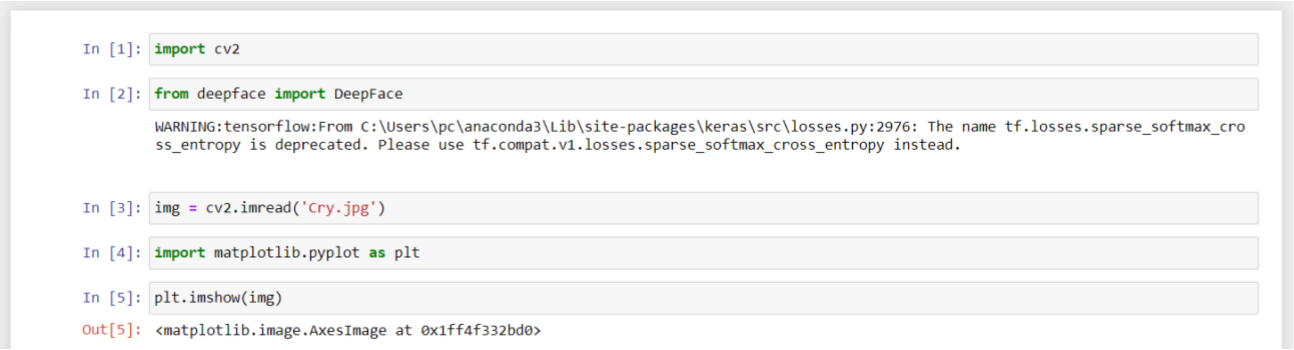
Facial Expression Recognition (FER) is a computer vision task aimed at identifying and categorizing emotional expressions depicted on a human face. The goal is to automate the process of determining emotions in real-time, by analyzing the various features of a face such as eyebrows, eyes, mouth, and other features, and mapping them to a set of emotions such as anger, fear, surprise, sadness and happiness.

Face emotional

Used Function & Libarary:

- ✓ cv2
- ✓ DeepFace
- ✓ matplotlib.pyplot as plt
- ✓ cv2.imread
- ✓ plt.imshow
- ✓ cv2.cvtColor
- ✓ DeepFace.analyze
- ✓ type()
- ✓ cv2.CascadeClassifier
- ✓ cv2.data.haarcascades
- ✓ cv2.COLOR_BGR2RGB
- ✓ cv2.COLOR_BGR2GRAY
- ✓ FaceCasCade.detectMultiScale

Section 1:



```
In [1]: import cv2

In [2]: from deepface import DeepFace

        WARNING:tensorflow:From C:\Users\pc\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cro
        ss_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

In [3]: img = cv2.imread('Cry.jpg')

In [4]: import matplotlib.pyplot as plt

In [5]: plt.imshow(img)
Out[5]: <matplotlib.image.AxesImage at 0x1ff4f332bd0>
```

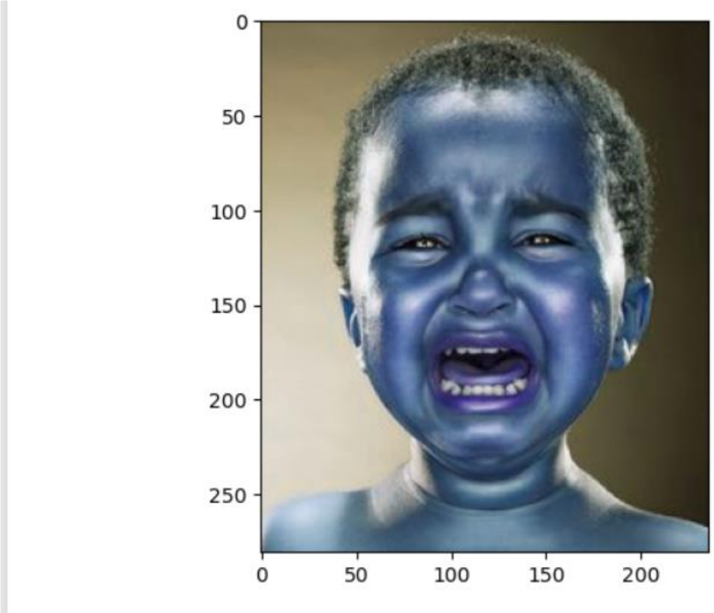
✓ **Import cv2 → which is a popular and powerful tool for computer vision, Image processing , face detection , object recognition, resize , preprocess the images.**

✓ **from deepface import DeepFace →we use It in Face recognition, Facial attribute analysis tool , pre-trained modesls . it used to access the emotion detection model.**

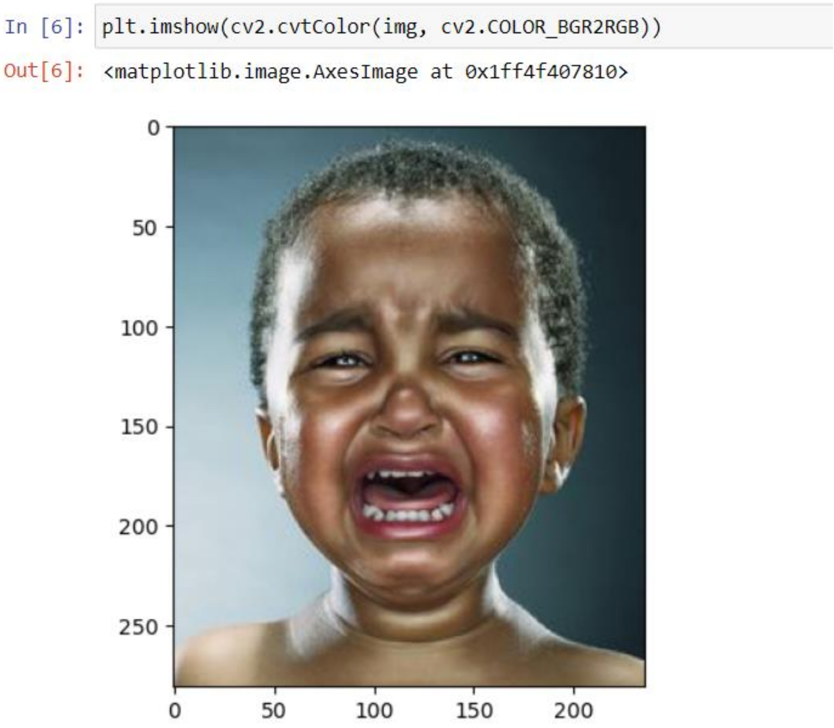
✓ **img = cv2.imread('cry.jpg') → read the image from the path.**

Import matplotlib.pyplot as plt → creating various types of plots and Graphs.

✓ **Plt.imshow(img) → display img as graph.**



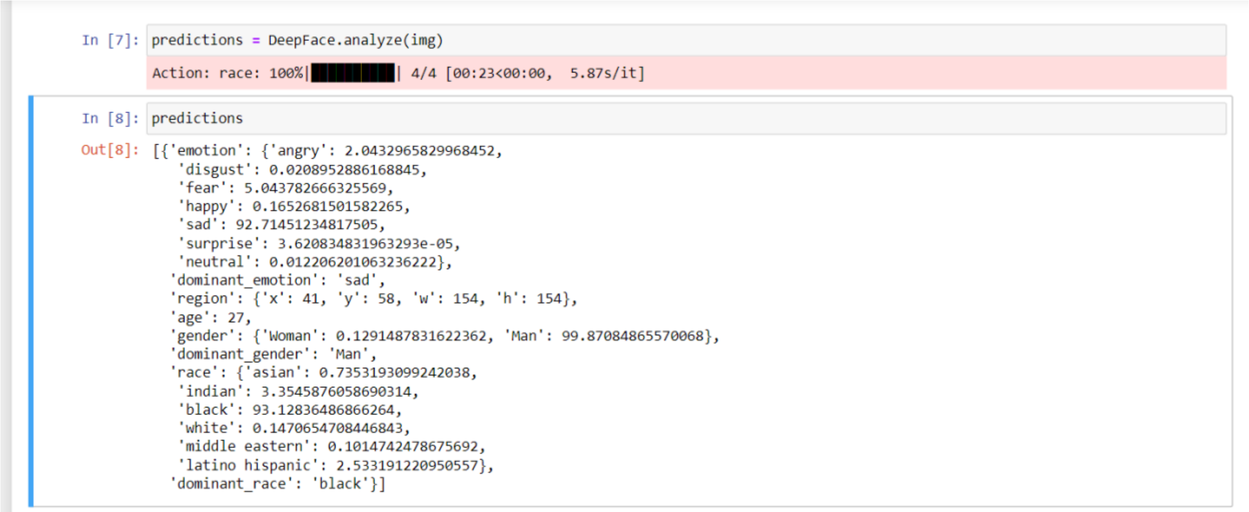
- ✓ **The axix refer to pixels in image , but the image with not correct colors!!**



```
In [6]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
Out[6]: <matplotlib.image.AxesImage at 0x1ff4f407810>
```

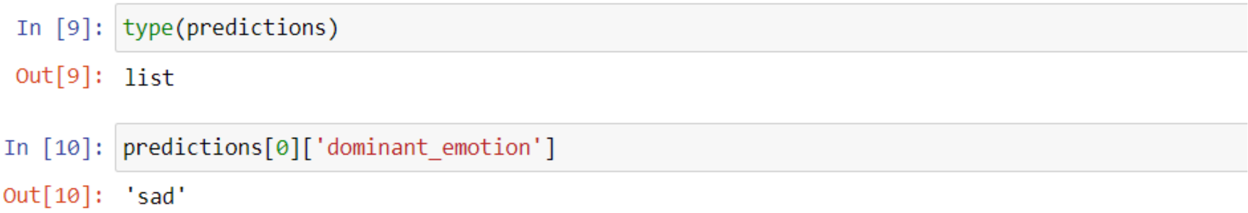
- ✓ **cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) → convert the image color to correct.**

Section2:



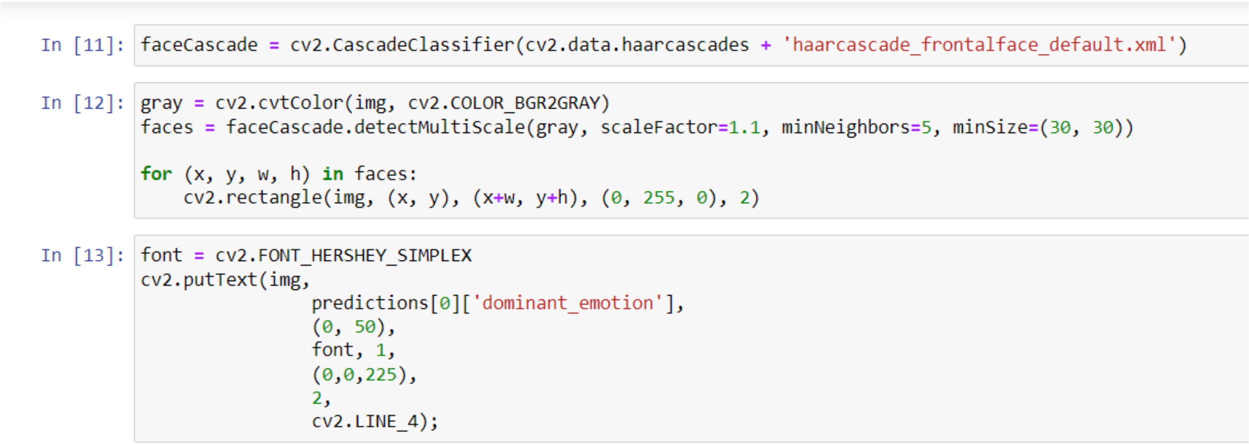
```
In [7]: predictions = DeepFace.analyze(img)

        Action: race: 100%|██████████| 4/4 [00:23<00:00,  5.87s/it]

In [8]: predictions

Out[8]: [{'emotion': {'angry': 2.0432965829968452,
           'disgust': 0.0208952886168845,
           'fear': 5.043782666325569,
           'happy': 0.1652681501582265,
           'sad': 92.71451234817505,
           'surprise': 3.620834831963293e-05,
           'neutral': 0.012206201063236222},
          'dominant_emotion': 'sad',
          'region': {'x': 41, 'y': 58, 'w': 154, 'h': 154},
          'age': 27,
          'gender': {'Woman': 0.1291487831622362, 'Man': 99.87084865570068},
          'dominant_gender': 'Man',
          'race': {'asian': 0.7353193099242038,
           'indian': 3.3545876058690314,
           'black': 93.12836486866264,
           'white': 0.1470654708446843,
           'middle eastern': 0.1014742478675692,
           'latino hispanic': 2.533191220950557},
          'dominant_race': 'black'}]
```

Predictions = DeepFace.analyze(img) → display all features of pre-trained model and save it predictions var.



```
In [9]: type(predictions)

Out[9]: list

In [10]: predictions[0]['dominant_emotion']

Out[10]: 'sad'
```

- ✓ **The type of features that model display is in list of direction format.**
- ✓ **List that displayed has two index, so we access the target emotion   from dominant_emotion.**



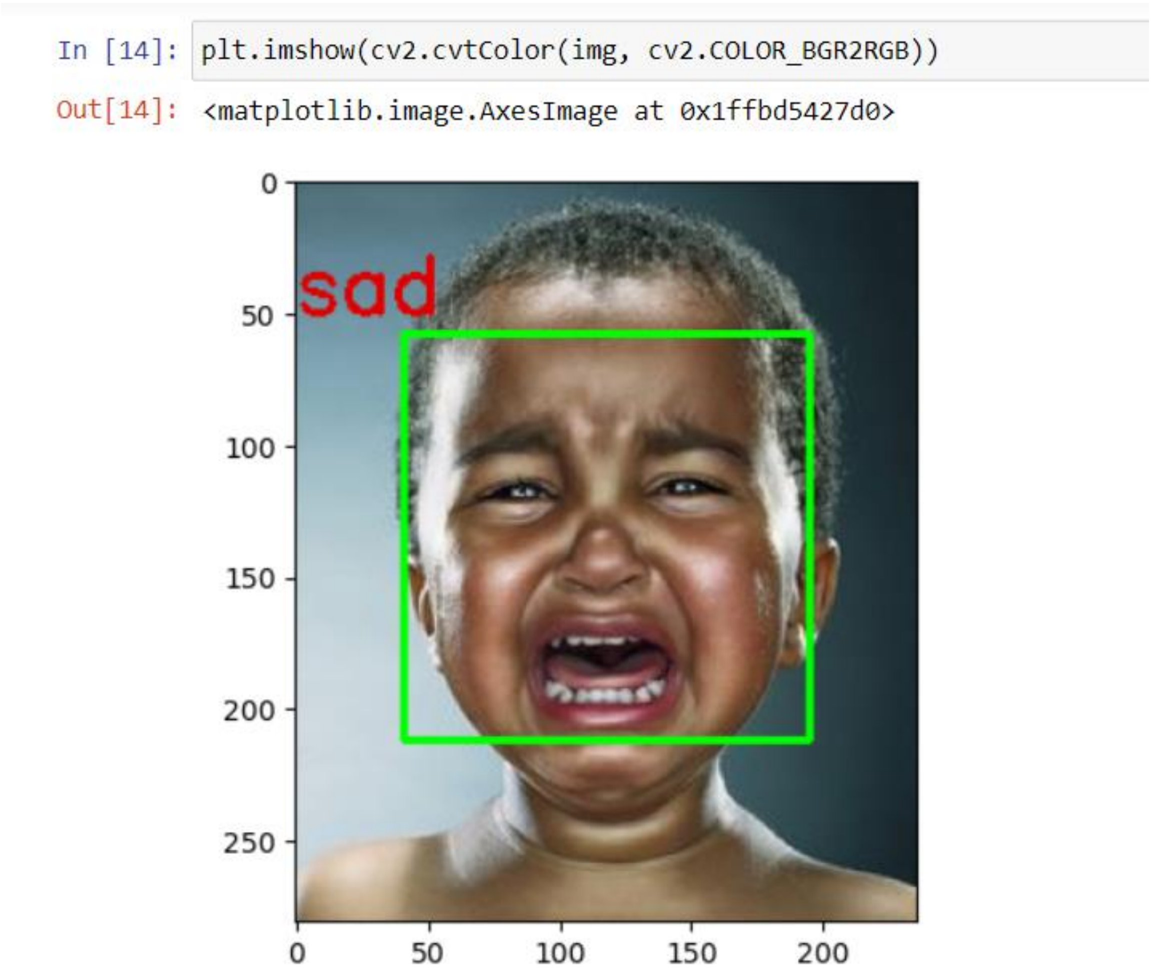
```
In [11]: faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

In [12]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

         for (x, y, w, h) in faces:
             cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

In [13]: font = cv2.FONT_HERSHEY_SIMPLEX
         cv2.putText(img,
                         predictions[0]['dominant_emotion'],
                         (0, 50),
                         font, 1,
                         (0,0,225),
                         2,
                         cv2.LINE_4);
```

✓ **We use cv2.CascadeClassifier as a model which has multiple files of detections so we ‘haarcascade_frontal_default.xml’ to detect face emotion and we get the path of this file using cv2.data.haarcascades.**

✓ **We convert the color of image to gray to reduce the complexity of image.**

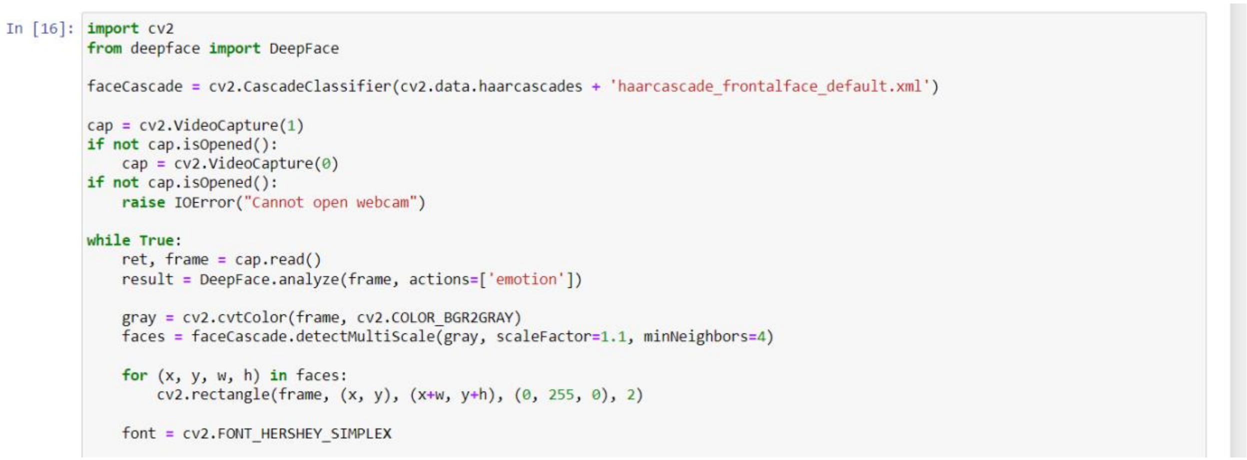
✓ **Next the faces store four value as scale of image X Y W H as X and Y refers to the top left corner of the image, W refers to width of image and H refers to height of image.**

✓ **Next we use function to draw rectangle in the detected face anfd display his emotion in the top left corner of the image.**

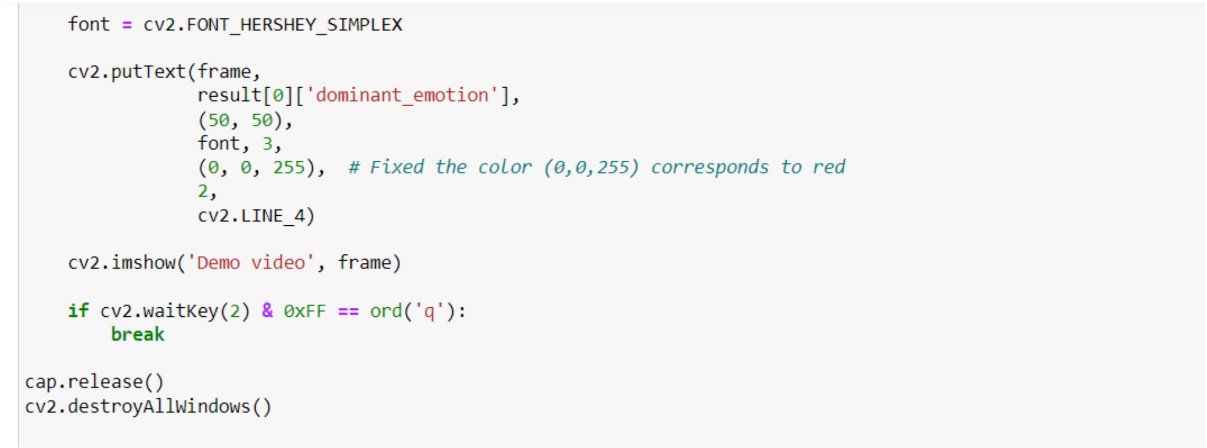


```
In [14]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[14]: <matplotlib.image.AxesImage at 0x1ffbd5427d0>
```

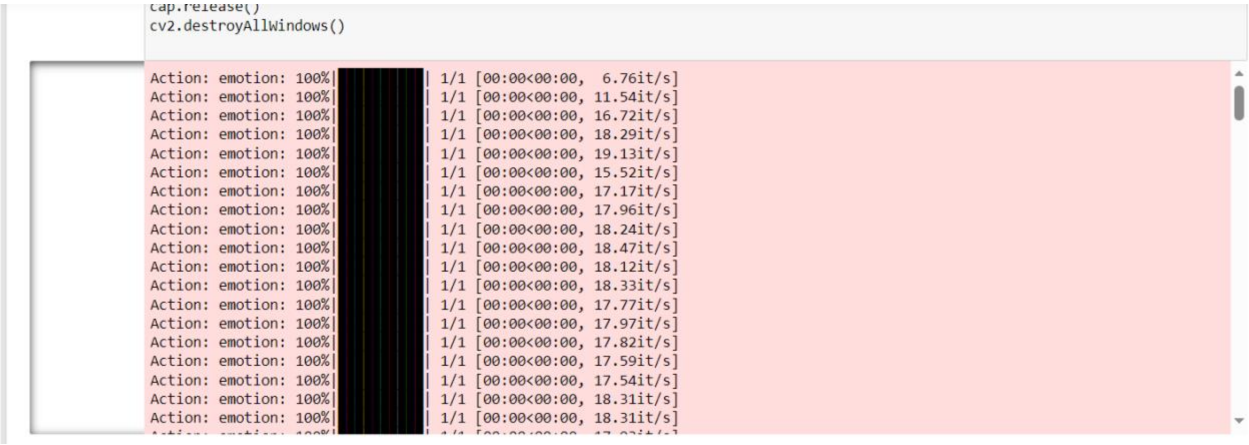
The last Section (WebCam):



```
In [16]: import cv2
         from deepface import DeepFace

         faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

         cap = cv2.VideoCapture(1)
         if not cap.isOpened():
             cap = cv2.VideoCapture(0)
         if not cap.isOpened():
             raise IOError("Cannot open webcam")

         while True:
             ret, frame = cap.read()
             result = DeepFace.analyze(frame, actions=['emotion'])

             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
             faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4)

             for (x, y, w, h) in faces:
                 cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

             font = cv2.FONT_HERSHEY_SIMPLEX
```

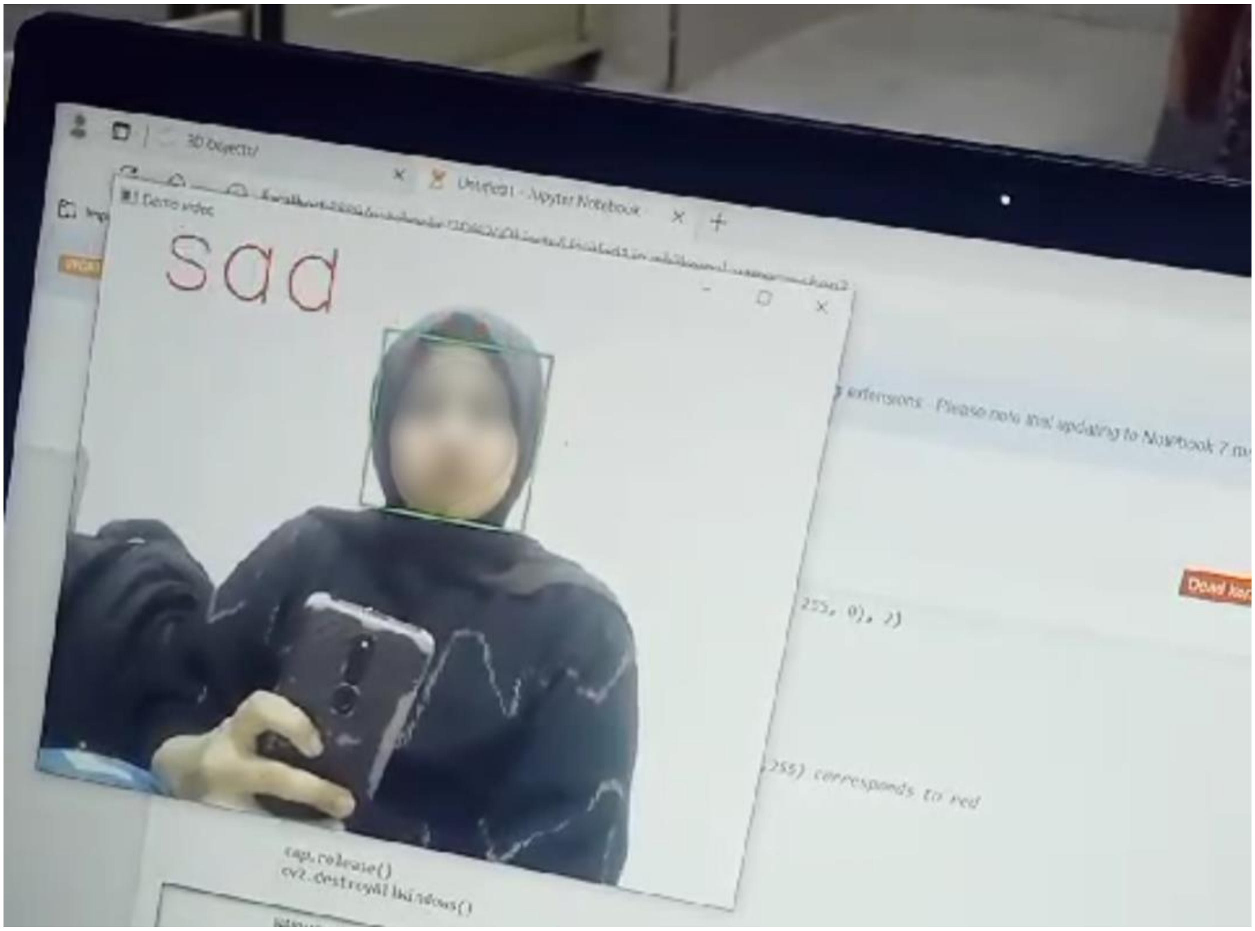


```
    font = cv2.FONT_HERSHEY_SIMPLEX

    cv2.putText(frame,
                result[0]['dominant_emotion'],
                (50, 50),
                font, 3,
                (0, 0, 255),  # Fixed the color (0,0,255) corresponds to red
                2,
                cv2.LINE_4)

    cv2.imshow('Demo video', frame)

    if cv2.waitKey(2) & 0xFF == ord('q'):
        break

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



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

```
Action: emotion: 100%|██████████| 1/1 [00:00<00:00,  6.76it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 11.54it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 16.72it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.29it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 19.13it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 15.52it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.17it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.96it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.24it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.47it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.12it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.33it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.77it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.97it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.82it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.59it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 17.54it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.31it/s]
Action: emotion: 100%|██████████| 1/1 [00:00<00:00, 18.31it/s]
```



extensions. Please note that updating to Notebook 7 m

```
255, 0), 2)
```

```
255) corresponds to red
```

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