Facial Emotion Recognizer on VGG16

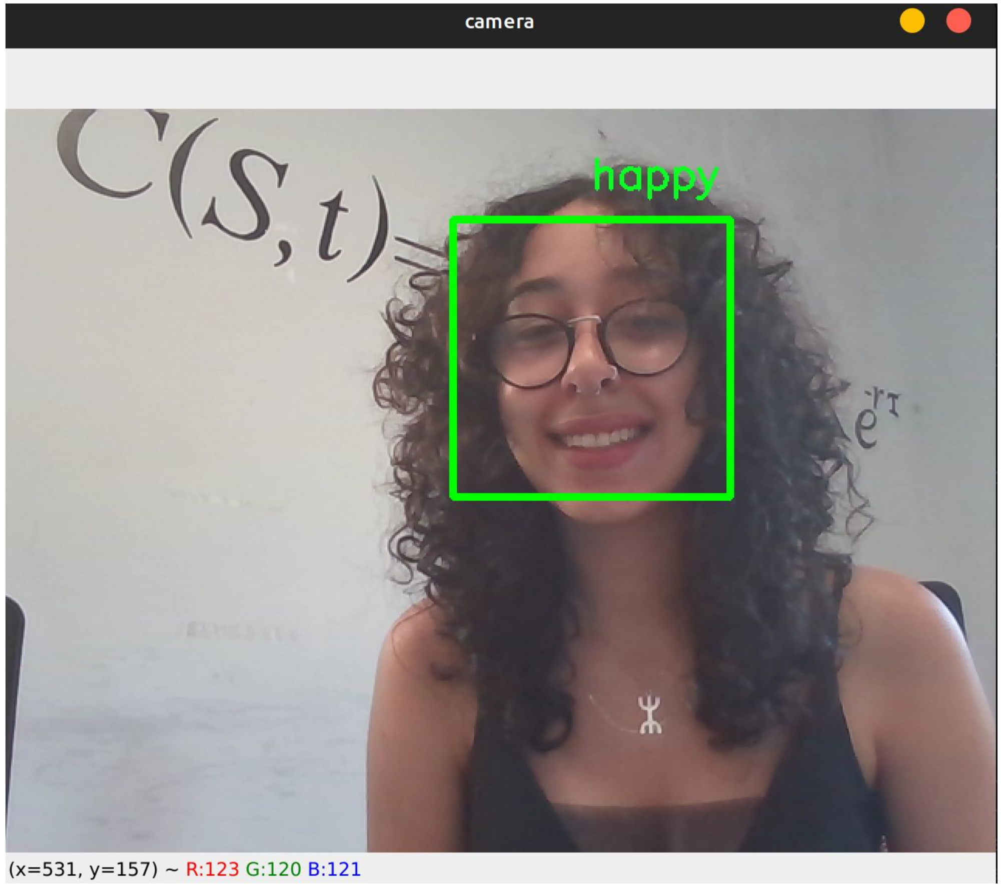
Francesco Fazzari, Gabriele Fabro, Diego Spaziani, Elisa Locchi

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

The first step was choosing a project .

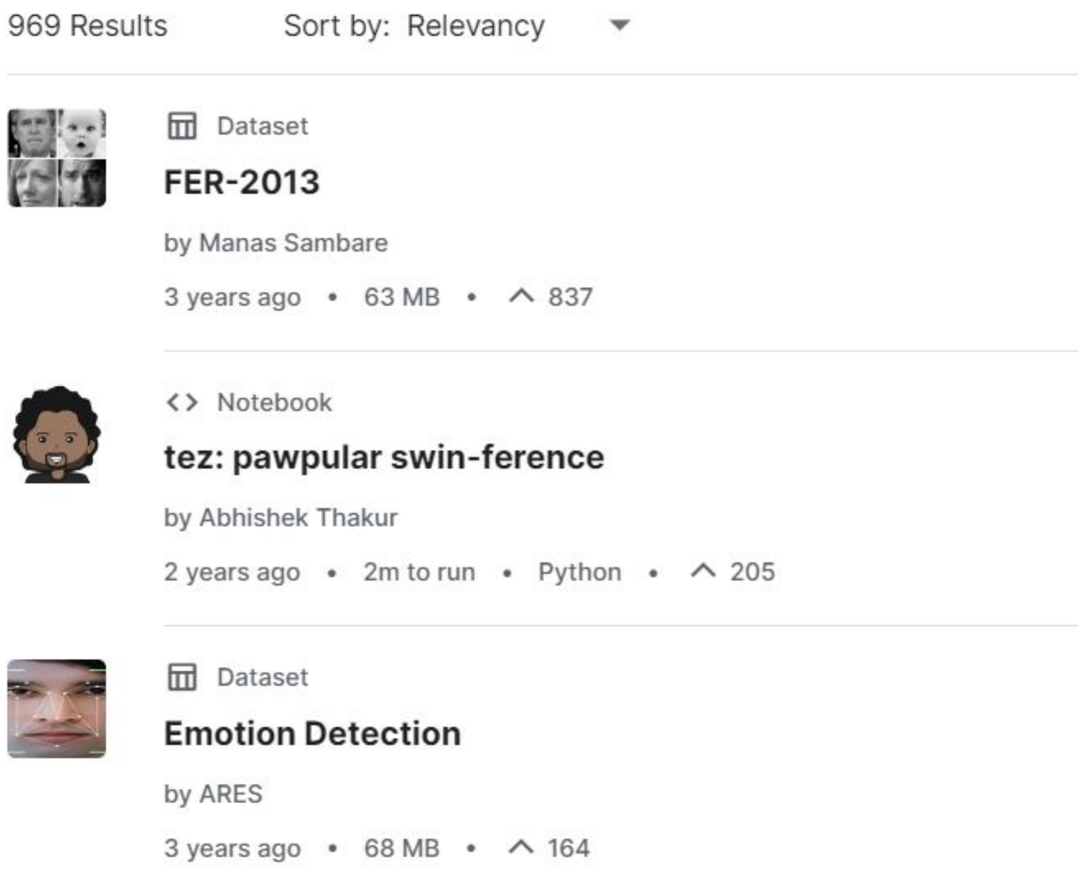
Surfing around the internet to search some ideas and datasets to use, we finally decided that our project would have been **FER: Facial Expression Recognizer**



Dataset

Ok, good idea, but now we have to find an appropriate dataset for our project!

Luckily it wasn't so hard: we found FER-2013, a dataset composed of more than 35k 48x48 pixel grayscale images of faces splitted in images for the train and for the tests.



969 Results   Sort by: Relevancy

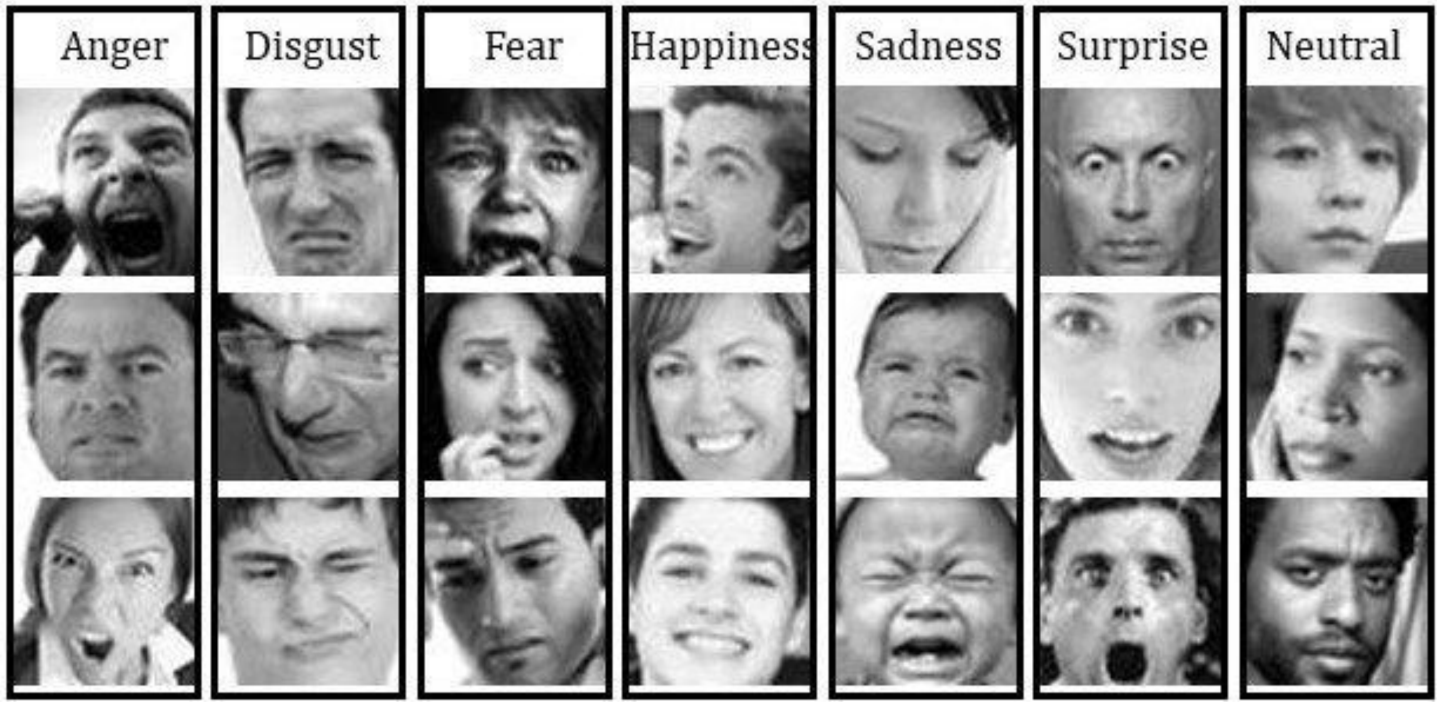
Dataset
FER-2013
by Manas Sambare
3 years ago • 63 MB • ^ 837

Notebook
tez: pawpular swin-ference
by Abhishek Thakur
2 years ago • 2m to run • Python • ^ 205

Dataset
Emotion Detection
by ARES
3 years ago • 68 MB • ^ 164

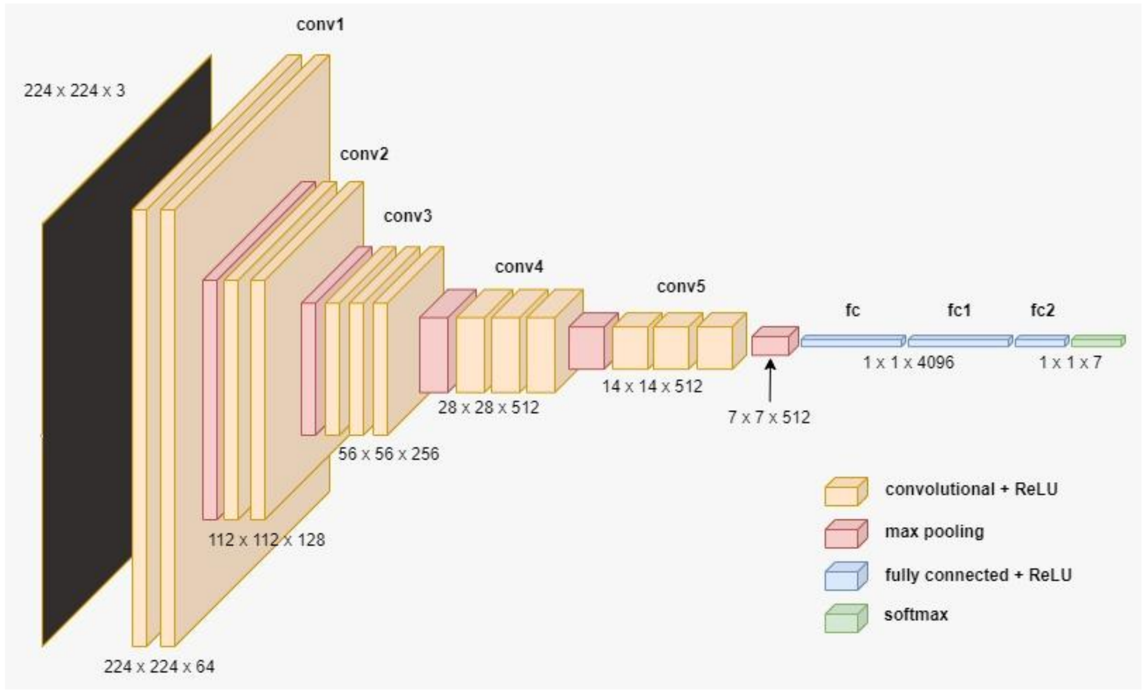
Dataset

The images are categorized in 7 different classes : Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral



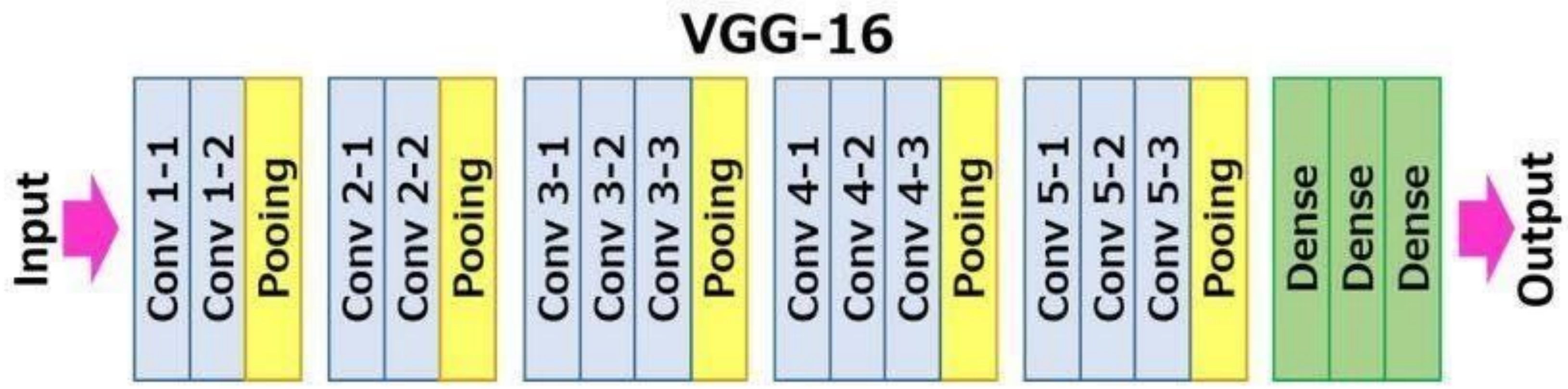
The Model: VGG16

It’s time to choose the model; the chosen one is the VGG16 due the great results while keeping the architecture simple. But how it works?



The Model: VGG16

The model takes in input the image which then goes through the different layers of the model where are performed operations like **convolutions**, **activations** and **pooling** .
The final layers of the model takes the features extracted from the previous one to map them to the output class.



The Adventure begins

The preparations are done, now we have to put all together. Obviously there were a lot of different problems.

First of all we have a dataset of 48x48x1 images while our model takes in input 224x224x3 images. Using transforms from torchvision we resized the images, then thanks PIL we converted our images in 3 channel RGB images.

The Adventure begins

Moreover in the dataset there wasn’t a validation set, so we just take 10% of the train set before start training the model, to build one

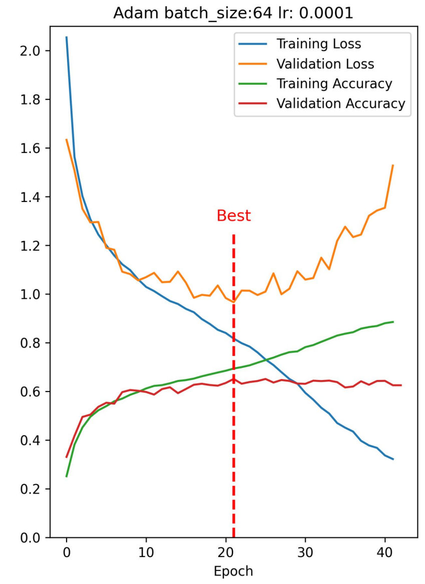
In the end, to avoid overfitting, we used RandomHorizontalFlip function from transforms on each images for every training

.. and continues

To train our model we tried a lot of different combinations of parameters and optimizer to choose in the end the one with the best performance. Here some results:

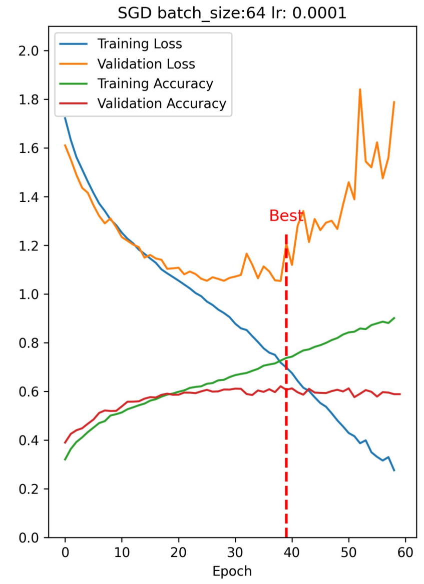
Best performance with:

- Opt: Adam
- Batch size: 64
- Lr: 0.0001



Best performance with:

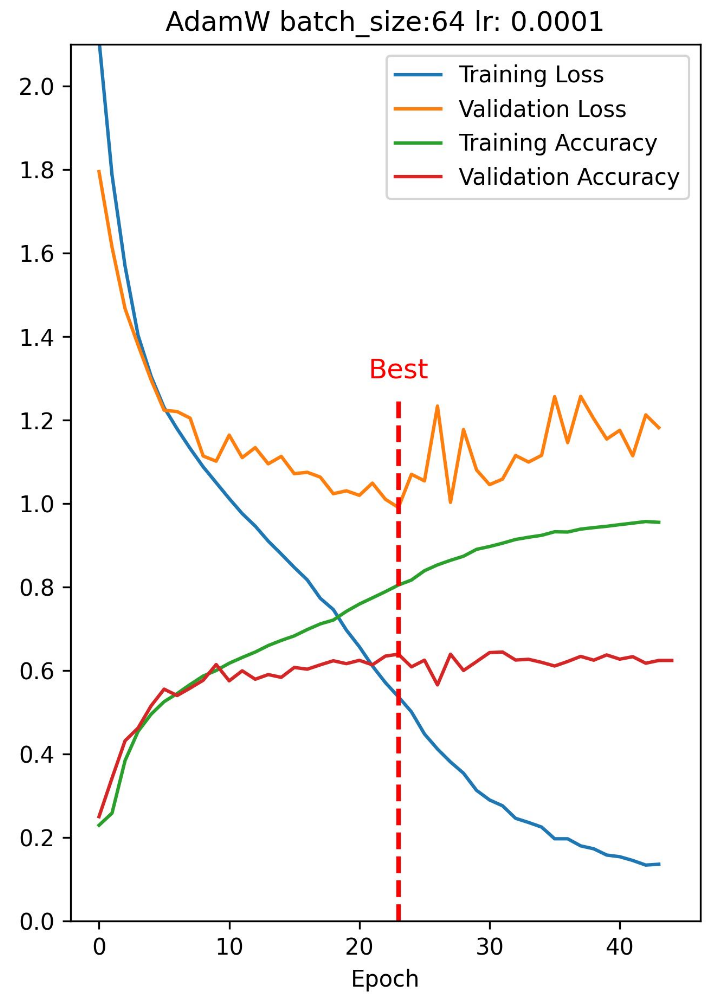
- Opt: SGD
- Batch size: 64
- Lr: 0.0001



Final choice

Our final choice as optimizer was AdamW, even though it gave us similar results compared to Adam due to its better management of the weight decay.

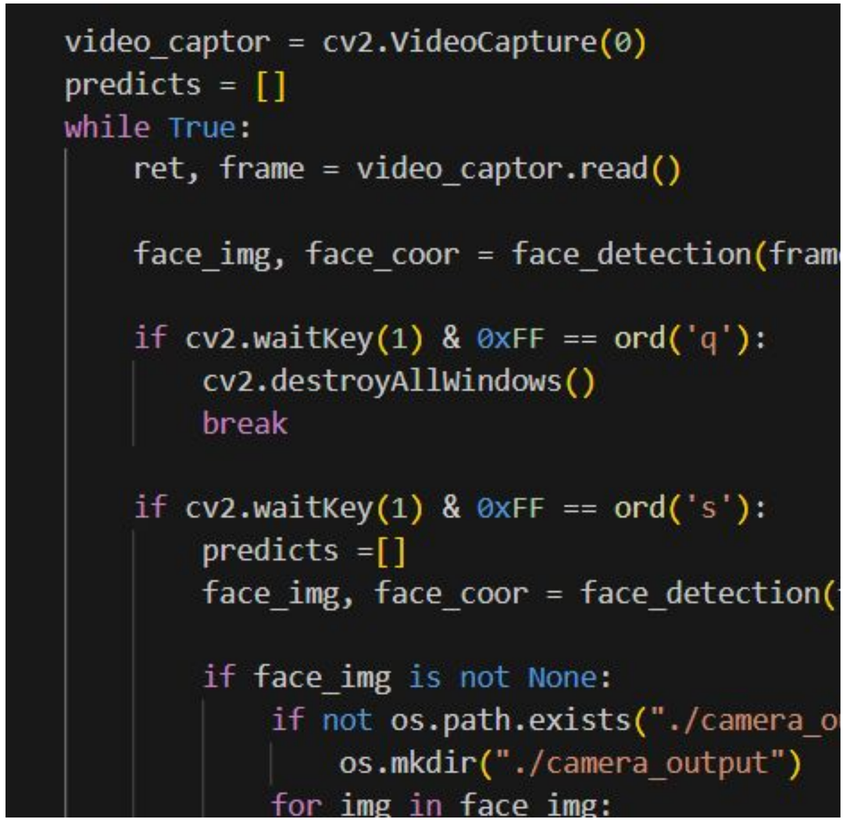
Final parameters:
Optimizer: AdamW
Loss function: CrossEntropyLoss
Epoch: 50
Early stopping: 20
Learning rate: 0.0001
Weight decay: 5x10^-4 compared to Adam due to its better management of the weight decay.



Icing on the cake

So in the end we decided to implement a simple feature to demonstrate our work in practice.

Icing on the cake

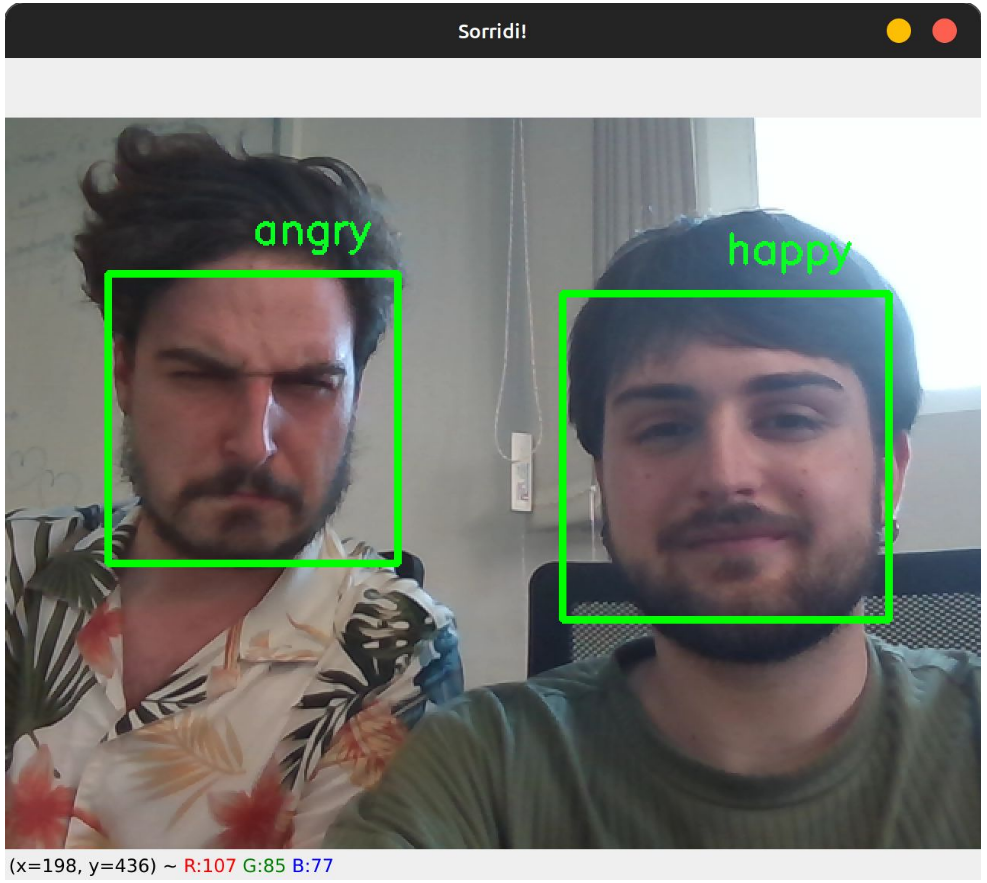


```
video_captor = cv2.VideoCapture(0)
predicts = []
while True:
    ret, frame = video_captor.read()

    face_img, face_coor = face_detection(fram

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

    if cv2.waitKey(1) & 0xFF == ord('s'):
        predicts =[]
        face_img, face_coor = face_detection(

        if face_img is not None:
            if not os.path.exists("./camera_o
                os.mkdir("./camera_output")
            for img in face_img:
```

Our script consists of taking frames from OpenCV's VideoCapture and pass them to a function of ours that takes advantage of the HaarCascades algorithm that recognizes faces and allows us to return a list of coordinates, which indicates the rectangle of the recognized portion of the face, and the image of the face cropped.

Icing on the cake

So we can pass the image to the model, with the coordinates shown in real time and the recognized portion of the face, via an interrupt the user can request the recognition of the emotion and eventually the result is shown on the screen above the recognized face.


Sorridi!
angry
happy
(x=198, y=436) ~ R:107 G:85 B:77