|  |
| --- |
| Image Processing |

-Face Filter & Mask–

측정기이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| Subject | Image Processing |
| Submit date | 2022.12.4 |
| Team | MYFJ |
| Name | 박지현/국연우/플로리앙 골링/압델말리크 아치테 헨니 |

Contents

1. **Outline of Program**
2. **Design of emotion detector & face filter**

2-0. AI model  
2-1. Angry filter  
2-2. Fear filter  
2-3. Happy filter  
2-4. Sad  
2-5. Surprise  
2-6. Neutral filter

1. **Design of mask modules (Bonus)**
2. **Design process of group discussion**
3. **Opinion**

**① Outline of program**

Our Program is detecting the picture’s emotion and applying filter according to the detected emotion. To recognize emotions, We used AI to train APIs and make AI to learn 6 different emotions: Happy, Angry, Surprised, Fear, Sad and Neutral. Filters are consisted of 2 parts. When we apply a image to program, program first detects the emotion and add a mask image to detected face . And then execute pixel operations to add filters to the whole image.

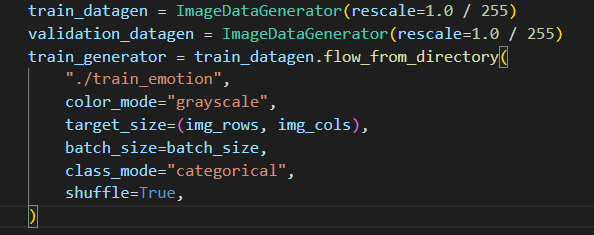
**② Design of program**

1. **Learning algorithm**

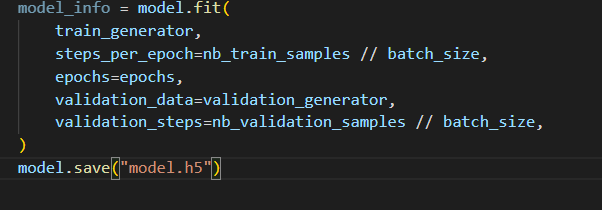
To find the most dominant emotion on the picture and the human’s face, we had to train an IA using TensorFlow. The first step was to find a great dataset allowing us to train the IA with test pictures, and validation data to tell the IA if its guess is good, or not.

We used a 28k+ pictures dataset and had every emotion in its own directory making it easier for TensorFlow to work with.

All pictures were to be reformatted in the program so the IA would train with samples of the same size, colors.



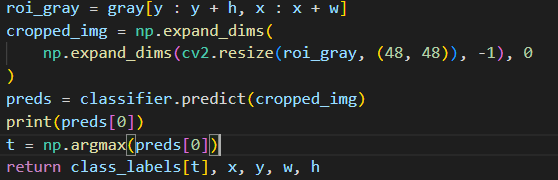
Then we just had to let it train and create a .h5 file containing the training data.

****

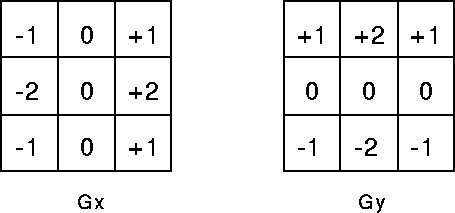
We added a feature to load existing model and train it even more instead of recreating a new model each time.

Once the model created, and its statistics good, we only had to load in into TensorFlow, reformat our input picture to make it like the sample images used in the training, and get the result according to its training.





1. **Angry filter**

텍스트이(가) 표시된 사진

자동 생성된 설명Angry filter makes image’s edge red and turns whole image even darker. To detect the edges, we made image gray and used sobel filter and convolution operation.

Table Sobel filter

These kernels return high response at edges running vertically and horizontally. After that, if a color value is higher than throughput 90 at each pixel, which means it’s around the edge at gray image, we designated the red as a high value, and other colors low to make edge red. To make image more dramatic, we subtracted 60 from all color so that red edge and dark ambient make us feel more emotional

Then we apply a glaussian blur filter twice on the image with a small mask size so the red lines added by the edge detection part will be smoother.

텍스트, 남자, 보는이(가) 표시된 사진

자동 생성된 설명**1.1 Result image**

Table Angry filter

1. **fear filter**

텍스트이(가) 표시된 사진

자동 생성된 설명We used lookup table to apply cooling filter for fear emotion. For creating lookup table, we used one-dimensional linear interpolation and make value of blue higher than red.

While the nested for loop, red has its value at location r of redLookupTable and blue has its value at location b of blueLookupTable. As a result, output image becomes more bluish than before.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2-1 Result image**

사람이(가) 표시된 사진

자동 생성된 설명

Table Fear filter

1. **Happy filter**

Happy filter divides image’s red, green, blue channel and apply it at a different range of location so that it looks dynamic and vibrating.

텍스트이(가) 표시된 사진

자동 생성된 설명First, we extract color value from 0 to before 100 of width size and apply only red value from 100 to width size to pixelToChange list. We used this list to emphasize color red of bottom left of the image.

텍스트이(가) 표시된 사진

자동 생성된 설명 Then extract color as the same way from 100 to width size and append only green value at from 0 to before 100 of width size to pixelToAdd list. We make new green value using this llist.

텍스트이(가) 표시된 사진

자동 생성된 설명And we add color blue of whole image to pixelAddBlue. /// I

* 1. 텍스트, 벡터그래픽이(가) 표시된 사진

     자동 생성된 설명**Result image**

1. **Sad filter**

Sad filter is simple in its algorithm, it will get the grayscale of the image an append a blur to it in order to make the overall mood of the image sad.

In addition, to enhance the sad face, we will only blur the background and all part of the picture which are not related to the face.

Une image contenant texte

Description générée automatiquement

**4-1 Result image**

Une image contenant personne, mur, intérieur

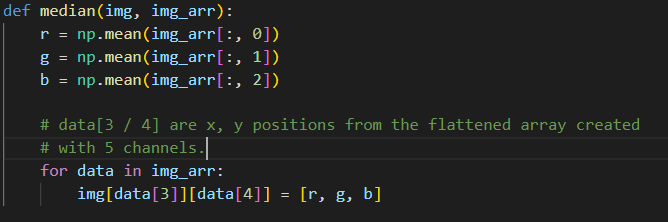
Description générée automatiquement

1. **Surprise filter**

We wanted to surprise filter to “cartoonize” the input picture. In order to create the effect, we must remove a lot of used colors in the picture and only keep a few to give the effect of drawing without a lot of shades.

The algorithm will look recursively for the most dominant colors of the pictures, and will change every colors close to them, into the same values as the most dominant, to remove the less dominant colors and only keep few colors.

The algorithm runs recursively 4 times in order to get output with enough colors to be recognized.



**5-1 Result image**

**Une image contenant texte, masque

Description générée automatiquement**

1. **Neutral filter**

We have no real ideas to what filter could link a neutral face, so we used it to make the picture smoother using contrast algorithm.

The algorithm will lookup for the smallest and biggest RGB value and will change every pixel according to it to make all the values closer and change the picture’s contrast making it smoother.

Une image contenant texte, écran, noir, argent

Description générée automatiquement

**6-1 Result image**

Une image contenant personne, mur, debout

Description générée automatiquement

**③ Mask Modules**

For the bonus part, we have one file per feeling. The principle is almost the same for each feeling. The goal is to apply a kind of filter on the image by adding an image on top. First we use the Mediapipe library to detect the face and then the parts of the face like eyes, nose and mouth. Then we recover the position in pixels of the parts of the face on the image. We then readjust the size of the image we want to place above the face according to the part of the face concerned. Finally we use a function to apply a kind of overlay on the original image to obtain the desired effect.

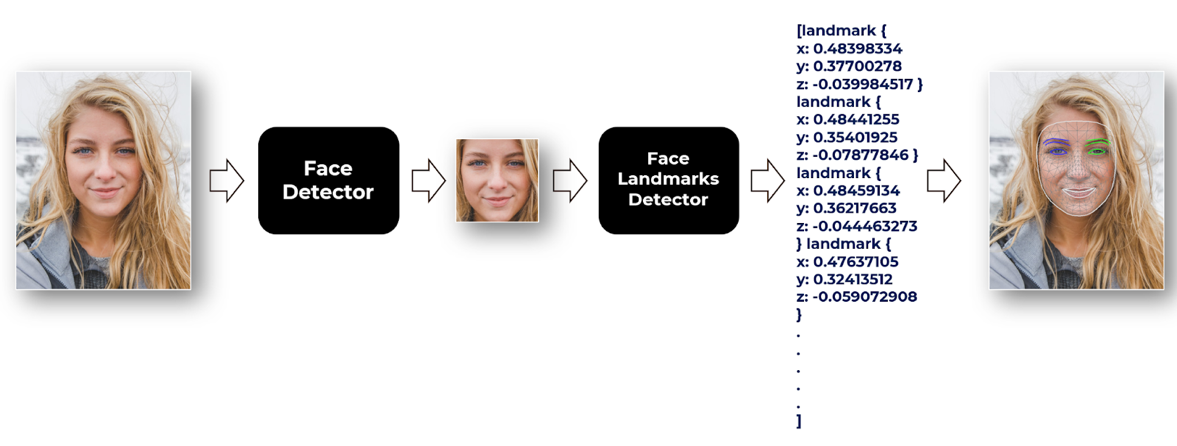
* **What is Facial landmark detection :**

Facial landmark detection is the process of detecting and tracking face key landmarks (that represent important regions of the face e.g, the center of the eyes, the nose, the mouth, etc in images and videos. It allows you to localize the face features and identify the shape and orientation of the face.

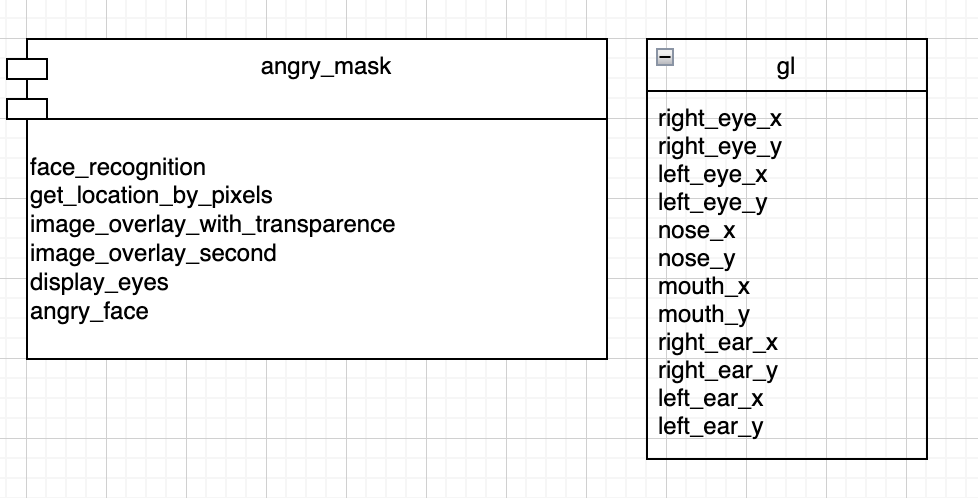
* **Use of Mediapipe to perform Facial landmark detection :**

MediaPipe is a cross-platform pipeline framework to build custom machine learning solutions for live and streaming media. The framework was open-sourced by Google.  
  
The machine learning pipeline of the Mediapipe’s solution contains two different models that work together :

1. A face detector that operates on the full image and locates the faces in the image
2. A face landmarks detector that operates only on those face locations and predicts the 3D facial landmarks.



* **Implementation of mask modules (angry\_mask, fear\_mask, happy\_mask, neutral\_mask, sad\_mask, surprise\_mask)**

In the angry\_mask module, several methods and class gl are defined for applying an angry mask to image as shown above.   
  
The Gl class has the position of each x and y pixel of the right eye, left eye, nose, mouth, right ear, and left ear as attributes.   
  
The face\_recognition function uses the mediapipe library to locate the positions of the right and left eyes, nose, mouth, right and left ears mentioned above with normalized pixel values and store them in gl.   
  
The get\_location\_by\_pixel function multiplies the normalized pixel positions by the width and height of the real image to convert them into the position of the real image, and stores them in gl.   
  
The image\_overlay\_with\_transparency function and image\_overlay\_second function are functions that transparently transform the rest of the mask image except for the specific area to be used for the mask and mask the specific location of the target image. The Image\_overlay\_with\_transparency function makes the background transparent for mask images with black background, and image\_overlay\_second function makes the background transparent for mask images with white background.  
  
In display\_eyes, the mask image is sized to fit the size of both eyes of the face, and the image\_overlay\_second function is used to mask the target image.   
  
In the angry\_face function, the target image is loaded, and a mask is applied to the target image using the above-mentioned functions.   
  
Since other mask modules (fear\_mask, happy\_mask, neutral\_mask, sad\_mask, surprise\_mask) have also been implemented through this logic, a detailed description thereof will be omitted.

**④ The project time line**

11/8

- First meeting to decide the subject  
- Made introduction PPT  
- Designed AI model and tried to modify it more accurately

11/10~11/25

- Developed each part and kept modifying it

- applied mask image for a bonus part

**⑤**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명 **Design process of group discussion**

**⑥ Team’s thoughts on this project**

Florian : The project in itself was pretty nice to do, learning about deep learning APIs using tenserflow was a first time, i was just surprised how many trainings and datasets it needed to be more precise. The hardest part was finding ideas for filters and getting clear and beautiful results

Malik : In my opinion the project was interesting in the way that we had the freedom to choose the subject as well as the technologies associated with the project. The project we chose was interesting because it combines artificial intelligence and image processing. Finally, I found out that Python was very adapted for image processing especially thanks to the various libraries available for it.

Yoenwoo : Through this project, I was able to understand libraries such as opencv and mediapipe more deeply and I could feel that image processing eventually means pixel-wise processing. In addition, I was able to improve my understanding by experiencing the process of generating an artificial intelligence model. As a result, I was able to further increase interest in image processing.

Jihyeon : I liked that we should have chosen the subject by ourselves. After we decided our subject as a face filter using Ai, the biggest challenge was that choosing what filter we were going to apply to the picture. We tried to use what we learned at our classes. I thought using Python was make things easier than what we did at our classes. Plus our teammates were so nice that I could learn more things by this project.