# Introduction

**Student Number: XXX**

This lab report is about face alignment and the lip/eye modifications. The face alignment is using Convolutional Neural Networks base on tensorFlow (with keras). The current algorithm is based on importing the processed data into a Convolutional Neural Network which with convolutional layer, pooling layer, dropout layer etc. And then use the predicted landmark from Convolutional Neural Networks model to change the color of eyes/ lips.

# Method

2.1 CNNs

2.1.1 **Preprocess data.**

(1) Converting each images in training data and test data to greyscale .And put all the greyscale images in two new array as new data sets. Compared to RGB images, the conversion to greyscale has two main implications: firstly, compared to RGB images, greyscale images take up less memory and are faster to compute; secondly, the conversion to greyscale increases the visual contrast and highlights feature.

(2)Resize each image in the new data sets after processing. from [244,244] to [96,96]. The processing carried out for the next reshape step. Without resize the images in the dataset, the total pixels would be much larger than the shape we want to reshape can handle.

(3)Reshape the new data sets, from[data. shape[0],244,244]( data. shape[0]:number of images in data set,[244,244]: greyscale images) to [-1,96,96,1].( -1: Data adaptation) Reshape can make the train and test run faster and use smaller memory. We also need to reshape the landmark. This is because the model outputs a one-dimensional array of coordinates of n images. The landmark as the target needs to be in the same format as the output. Otherwise the weights cannot be calculated and updated.New shape of landmark :[ ,,…, , ].(i is the number of landmark)

2.1.2 **Train and save model**

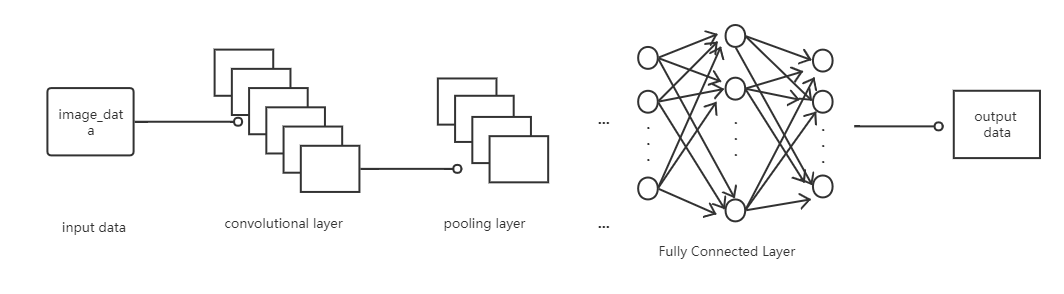


Figure 1 The model we build for solving the Face alignment.

In this model,use keras method power by TensorFlow to build the CNN model. Because keras can easier to build a CNN model. The following is introduce the important layers and parameters in CNN.

(1) Convolutional and pooling layers

Convolution being similar to feature extraction in image processing, and pooling being very similar to dimensionality reduction, with maximum pooling and average pooling being commonly used.

The image features are extracted by conv2D and compared to the true landmark using the activation function LeakyRelu and then the weights are updated. The number of convolutional kernels is incremented from 32 to 512, each time the number of kernels is increased, the features of the previous layer can be extracted more fully.

The pool\_size = (2,2),because after convolution, the x or y coordinates of the predicted feature points are obtained. It is a single number, so we use (2,2)

(2)Fully Connected Layer, Dropout, and Flatten.

Dropout is used to randomly discard some neurons with a certain probability during training to obtain higher training speed and to prevent overfitting.

Flatten is used between the convolutional layer and the fully connected layer to flatten the multi-dimensional data output from the convolution into one-dimensional data for the fully connected layer (similar to the shape method).

(3)optimizer ,loss and fit

Adam is able to adjust different learning rates for each different parameter, updating frequently changing parameters in smaller steps and sparse parameters in larger steps. Can converge more effectively. Loss:mean\_squared\_error. The fit function returns a History object, whose History.history property records the values of the loss function and other metrics over epoch. Batch size specifies the

number of samples to be included in each batch when performing gradient descent. The larger the batch size, the more batches per epoch, the faster it is, but the less accurate it is. Finally save our trained model.

2.1.3 Test data result

Import the test set data into the already trained model.. And reshape the result to [-1.42.2].Because the result is [test\_data.shape[0],84],it can not import into the function save\_as\_csv().And then read the csv, place the landmark of the processed csv on the corresponding test set image.(figure2)

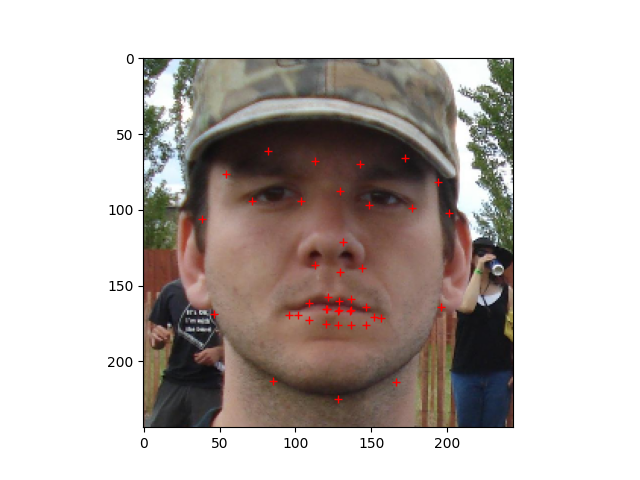
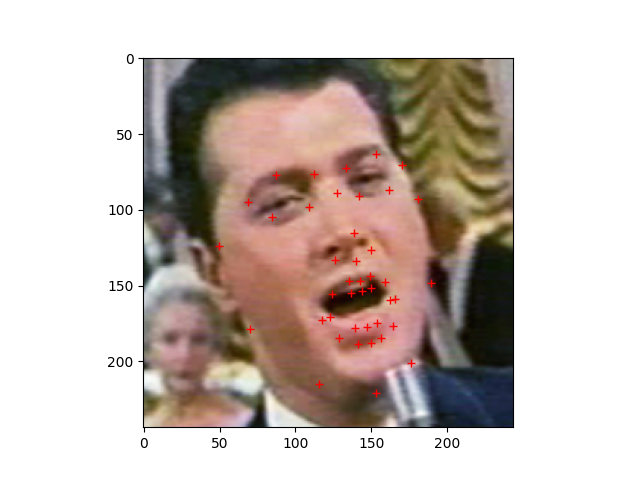


Figure 2

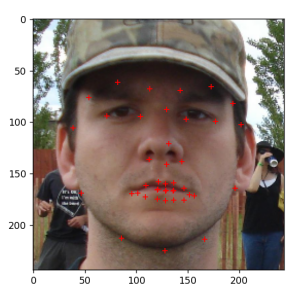
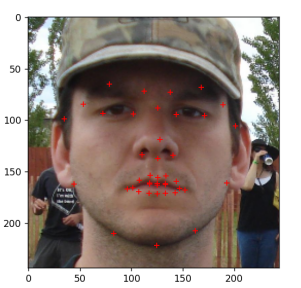
Different images in 2000 epochs model (difference face pose)(figure3)

图形用户界面, 应用程序

描述已自动生成图形用户界面, 应用程序

描述已自动生成Figure 3

Different results for the same image in 200 and 2000 epochs.(figure4 ,left image is 2000 epochs, right image is 200 epochs)



2.3 lip/eye modifications

This section is performed in two parts, the eyes and the lips. And through the loop , we can know that the index 0-6 is Contour of the face\7-9 is Left eyebrow \10-12 is Right eyebrow\13-17 is SNose\18-19 is Left eye\20-21 is Right eye\22-41 is Mouth

2.3.1 eyes modifications

First we can obtain the landmark of our predicted. And the eyes index is 18,19,20,21.

The following use the left eye as an example to explain. Left eye index is 18,19.so we can get this two eye points’ Coordinate position .18point name as A ,19point name as B. Calculate the length of the eye from the two points AB name as AB\_len(black line in figure 5(1)) ,and Calculate the width of the eye from the two points AB name as AB\_Width(red line in figure 5(1)). By these four figures we can calculate the center point C.(C = [A[0]+AB\_len/2, A[1]+AB\_width/2].(figure 5(2)) The y-coordinate at point C is added or subtracted AB\_width/2. Then we can get point D and point E.(figure5(3)).In the first time, I just using this 4 point (A,B,D,E). Putting A D B E in the a list name as mask\_list. And then by use cv2.fillPoly() to get a mask and change color. And then use cv2.addweight() blending translucent masks with images.(figure 6)图表, 散点图

描述已自动生成

(figure 5 Flow chart for calculating all coordinate points of the eye)

(figure 6)

Afterwards I added 4 points to the mask list (figure7, (F,G,H,I)) to make the mask more similar to the shape of the eye by add some points.(figure8). I also achieved changing only the corneal color by using the cv2.circle function. The parameters needed for this function have been calculated above.

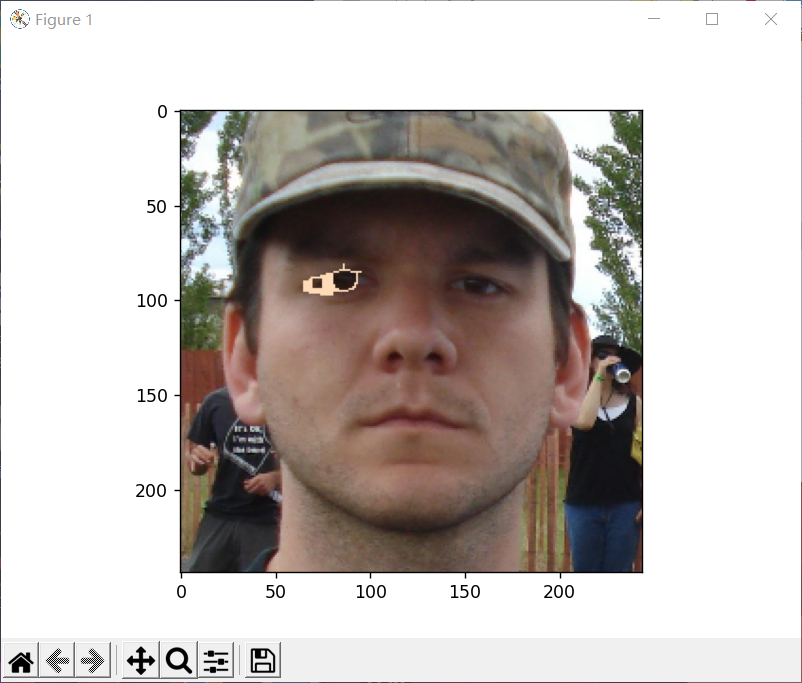
图表

描述已自动生成

Figure7

Before I implemented the above method. I had used a different method to implement it. The result was not very good.

In this method, I first calculate a bounding box based on the landmarks of the eye and cut the image according to the bounding box. Then, I use the canny operator to detect the edges(figure10(1). Then using morphological processing. Swell first, then corrode for closed operations(figure10(2). The processed template is then blended with the original image. The result is an eye color change. Here I have only done the implementation of the left eye.(figure10(3))



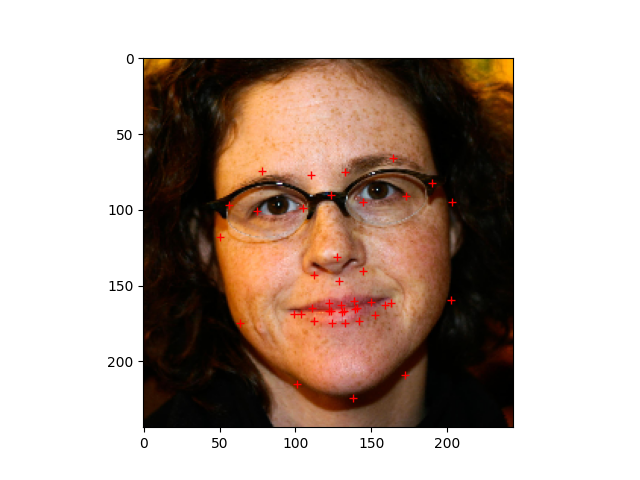
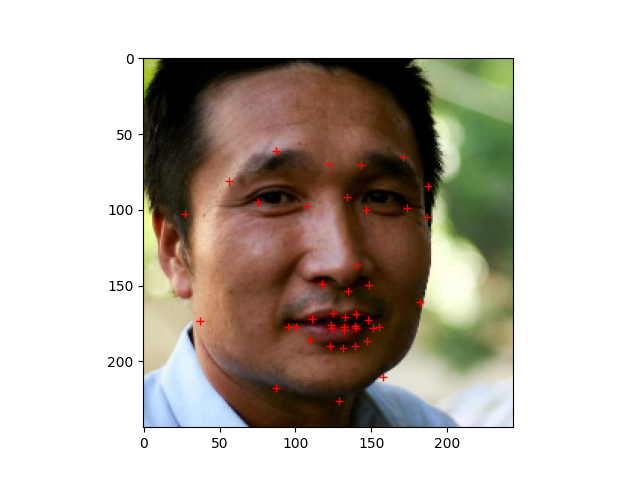
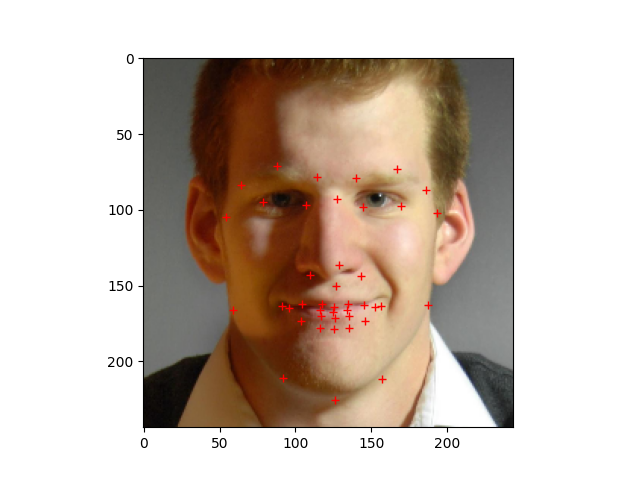
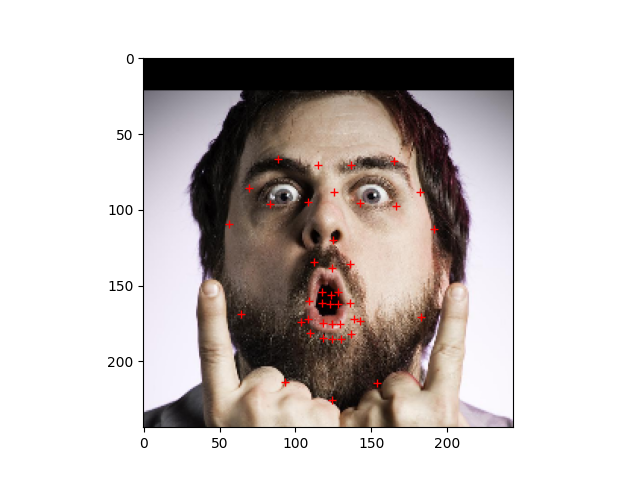
(1) (2) (3)

Figure10

2.3.2 lips modifications

The method of this section is similar to that of the eye. I have divided the lip section into up and down lips. Here is an example of the up lip and the same for the down lip. First of all, we know by traversing the landmark that the indexes of the up lip are [22:28], [35:38], So put these indexed points in the lip mark list. And then by use cv2.fillPoly() to get a mask and change color. And then use cv2.addweight() blending translucent masks with images.

3 Result

3.1 Example data set’s out put face alignment images(figure 11)

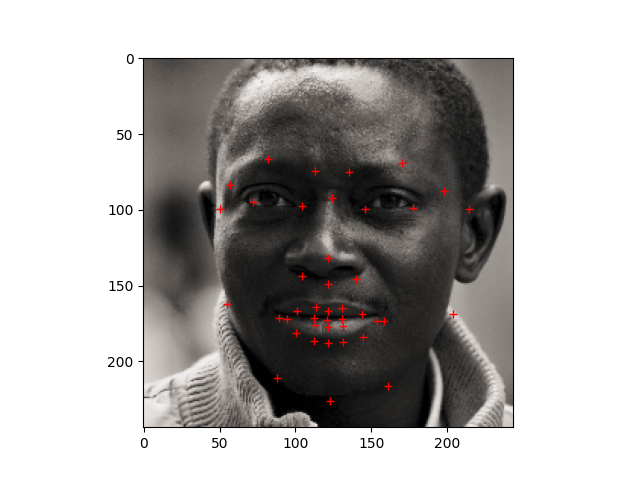
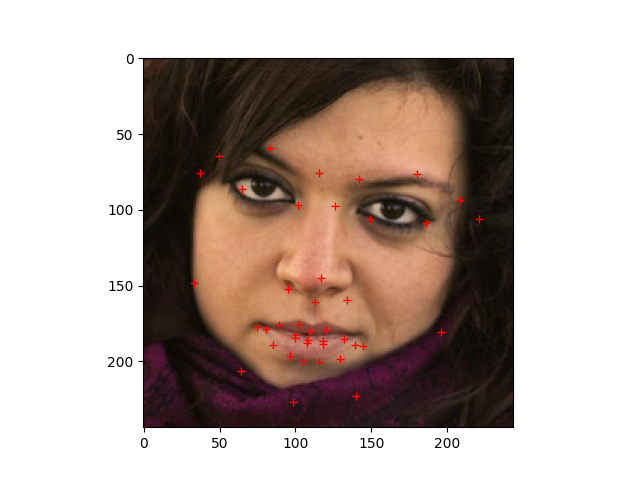
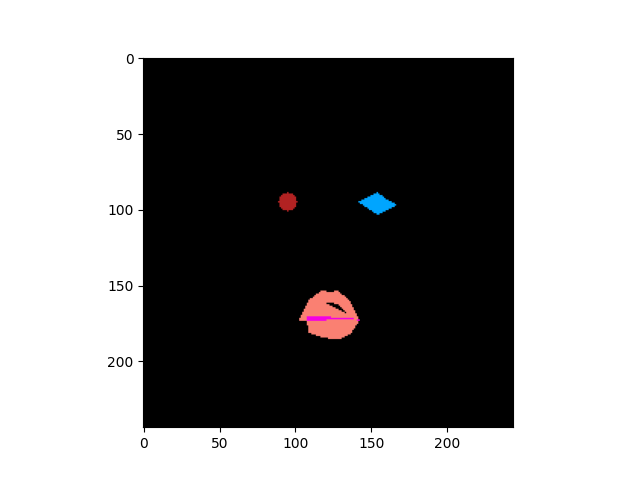
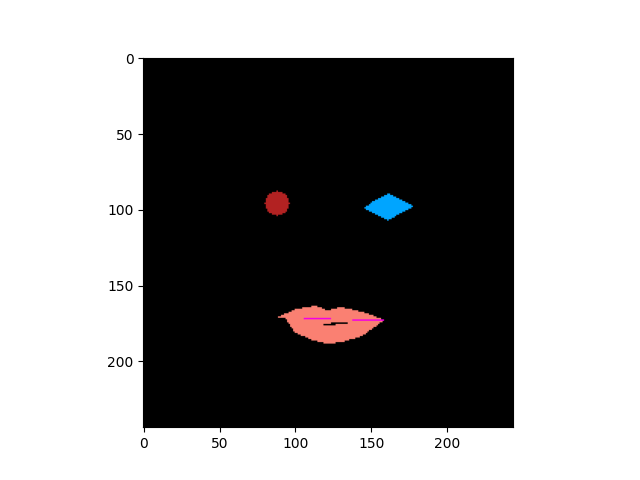
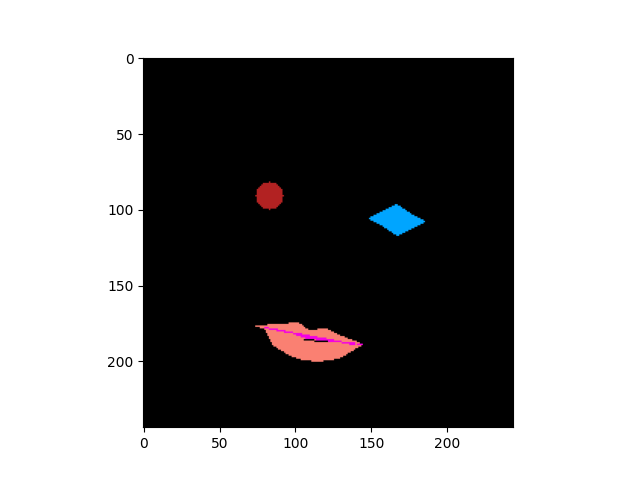
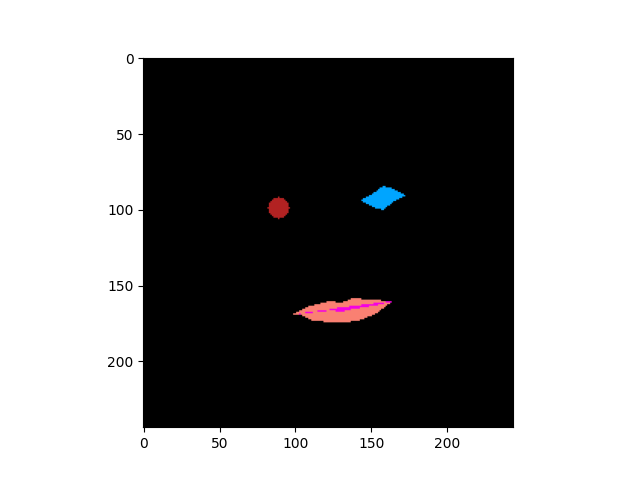


Figure11

It can be seen that the model gives good results when do faces alignment for different genders, races and poses.

3.2 eyes/lips modifications

3.2.1Mask image of eyes and lips modifications(figure 12)



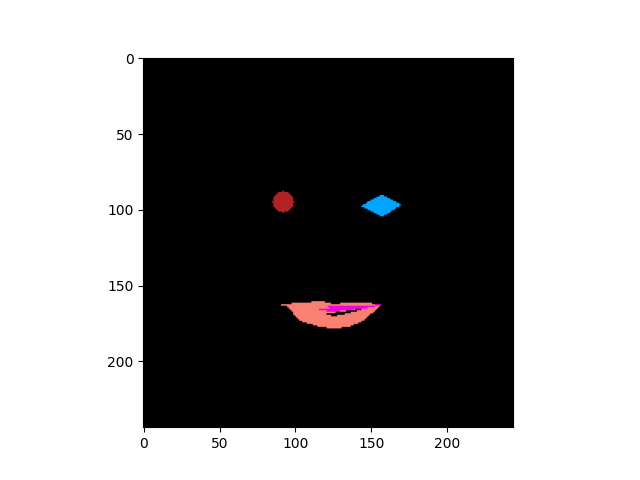
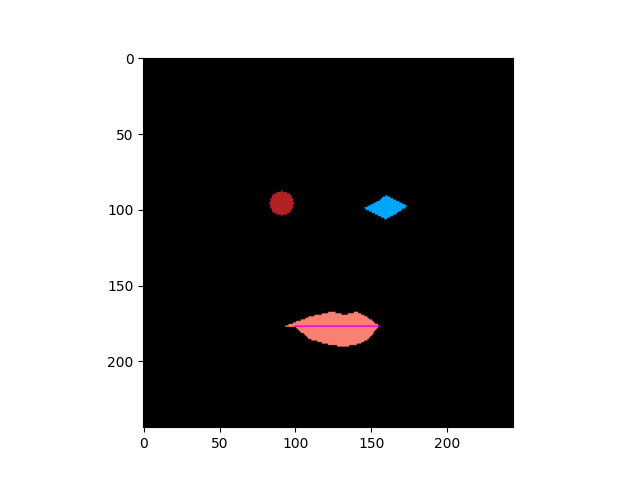
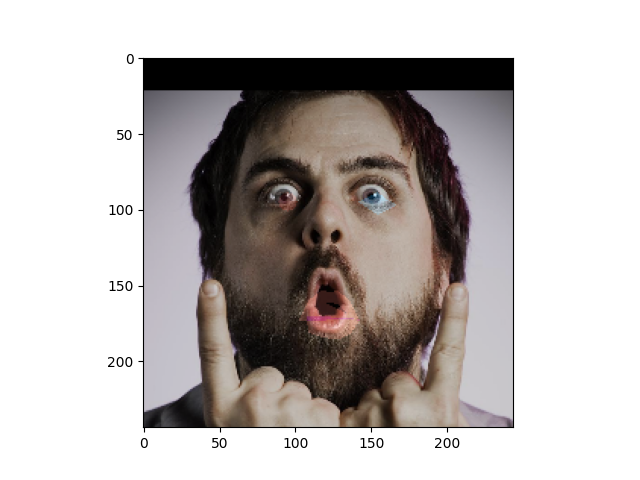
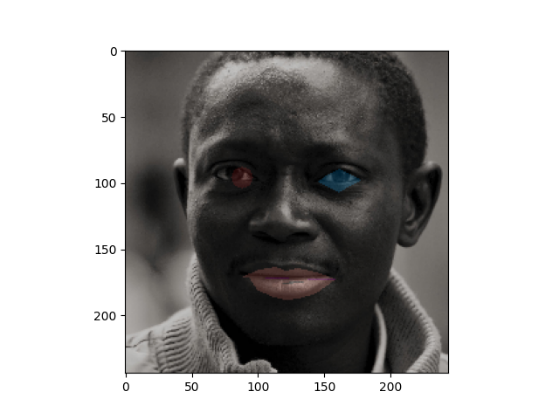
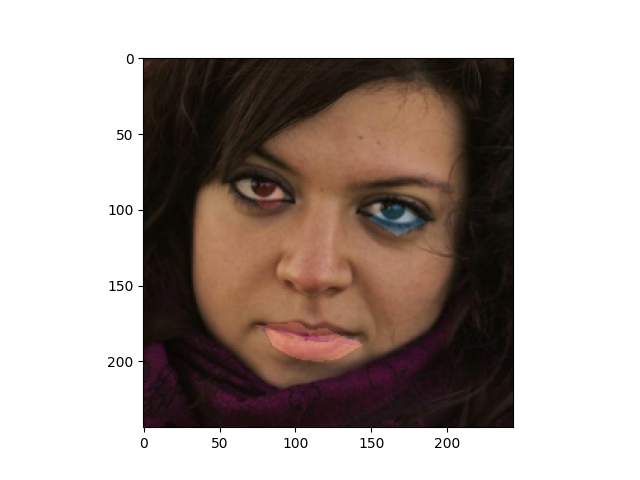
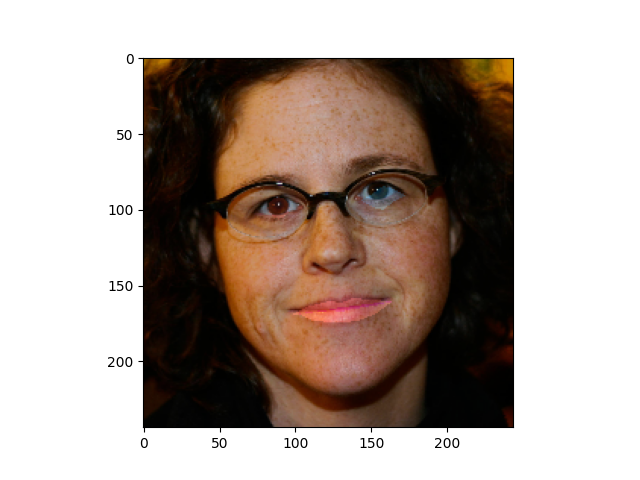


Figure12

3.2.2 eyes and lips modifications result(figure13)



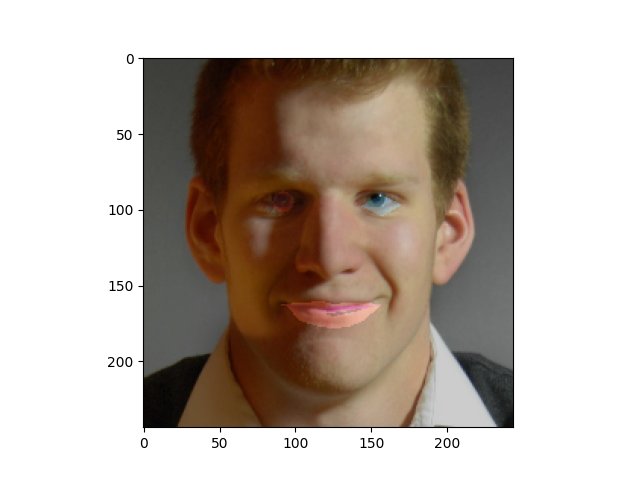
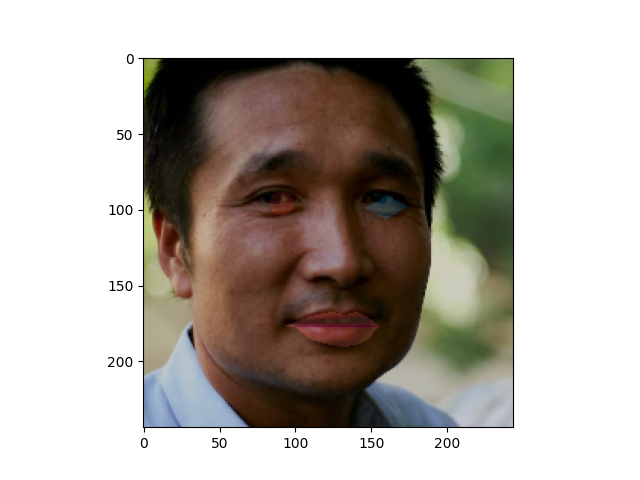


Figure13

Left eyes just change the corneal color, and right eyes change full eyes area color.Through the result , the Accuracy of Strange expressions is lower.

4 Reference

[1] Yüksel, K. and Skarbek, W. (2019) ‘Convolutional and Recurrent Neural Networks for Face Image Analysis’, Foundations of computing and decision sciences, 44(3), pp. 331–347. doi:10.2478/fcds-2019-0017.

[2] code by bruceyang2012 ‘s github(2018) <https://github.com/bruceyang2012/Face-Alignment-with-simple-cnn>

[3] <https://keras.io/>

[4] https://www.tensorflow.org/