Project Report

Introduction and related work

We aim to build a video face detection method, which can locate the position of human faces and recognize the gender, race and age for people who appear in the video. Bace on this method, we tried our implementation of face swapping. We pick this since face recognition and swapping is a trending thing nowadays and it also needs to apply neural network and deep learning.

We have discussed for a long time how to implement face recognition. We searched for simple application of face detection and recognition to get a point of how to implement it. After we have a point of how to implement detection method, we started to consider how to build the neural network. We used the experience of building neural network in a3 to build the network for our recognition method. We expected that the same network will work for our recognition method, but it turned out it is totally not the case. So we have to build a new one for our estimation of gender, race and age. We expected the main challenge is to build the network and find a proper dataset that can train our model. The limitation of our existing model is that the age an integer from 0 to 100, while the gender is binary 0 or 1, and the race is five integer 0, 1, 2, 3, 4. Unlikely from the previous assignment, we need to get three number x, y, and radius. We thought that we could not use the same loss function for each argument.

However, before we built the neural network, we were think about how we need to predict the information of people appeared in a video. I consult some papers about face recognition and swapping to find out what they predict. We concluded three main factors that needs to be predicted, which are gender, race and age. I found several famous face database like IMDB-WIKI – 500k+ face images with age and gender labels[1], Labeled Faces in the Wild[2], and color FERET Database[3]. But their label is not easy to treat and extract. Finally we found UTKFace[4], which contains more than 23000 human faces with labels explicitly shown in file name. The 0 and 1 stands for gender, and race is integer from 0 to 4, stands for White, Black, Asian, Indian and Others and age is integer from 0 to 116. We chose to use this dataset since it provides various human faces and clear labels.

Methods

1. We tear apart the video into frames and get a list of pictures. For a single frame, we used cv2 face detection to get a list of faces. And we put our faces into our model with trained weights to get a prediction of gender, race and age.
We encountered our first problem, how to deal with the training data. Since there are less cases that people over 100 shows in a video, and also for reduce the size of

dataset, I deleted images with people from 101 to 116. Then I separated the dataset into training set of size 13000 and validation set of size 4600 following a test set of size 4300. It took me 4 hours to upload these data, but when I try to load these data, it still took me more than 2 hour to load training data using `cv2.imread`, I knew that I had to change the way I store these data. At first, I wanted to keep all the image `np.array` and its label in a tuple and write them into `.txt` file, but I found it hard to read them and convert them into `np.array` and tuple of integer. So I found an effective way to save the np.array. I put the all image array into a big array whose size is (n, 200, 200) or (n, 200, 200, 3) depending on how I want our data to be grayscale or rgb. And I used the numpy built save to save these array into `.npy` file. For 13000 grayscale pictures, it generated a 5GB file to store them. However, the image information file for rgb pictures was too large to save, the colab would crash if we tried to save 13000 rgb file. So we decreased the rgb training, validation, test set size to 6000, 2000 and 2000, and save them into another google drive. I put the labels into `.txt` file and put them together with its corresponding `.npy` file. After this processing of data, everytime I loaded training, validation and test set only took 1 to 2 minutes, which is a huge saving of time.

2. We built the neural network ourselves, and tried three main different structures, and created several models by modifying the input and output data structure. For the neural network model, we first tried one that rescale age from 0 and 1 to 0 and 100, and race (0, 1, 2, 3, 4) to (0, 20, 40, 60, 80), which is same scale as age (0 to 100), using MSE as loss function. The advantage is that this method runs fast, while the disadvantage is that it could not give us a precise outcome.
The second model is that when we train our model, we divide our network into three branches. Each deals with one parameter, and uses categorical cross entropy as loss function.

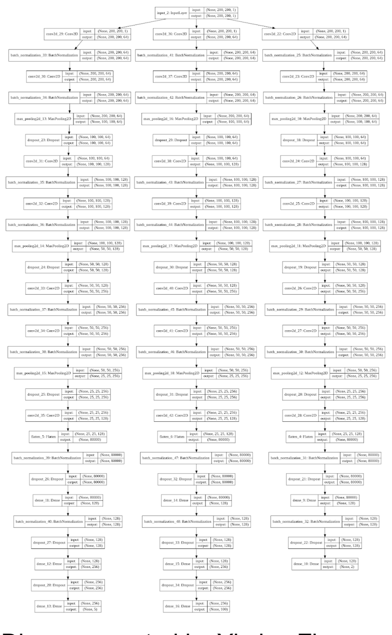


Diagram created by Yiming Zhang

The third model is that while we predict race and age, we used one branch for convolution layers and two different fully connected layers respectively. There is an individual branch for predicting gender. All these used categorical cross entropy as loss function. We used Mean square error for training age. We also created a RGB version.

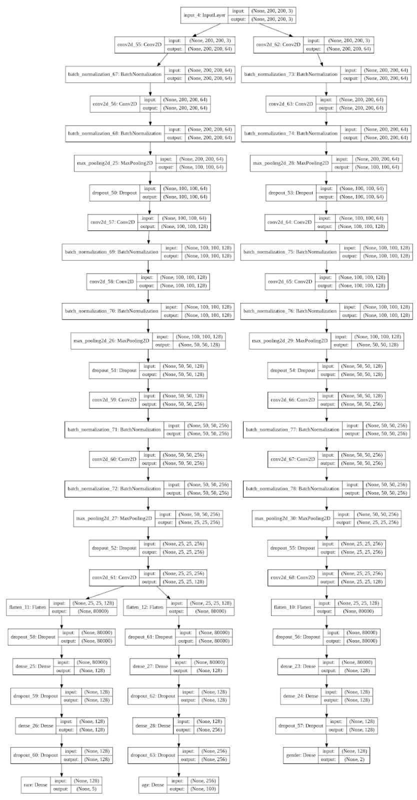


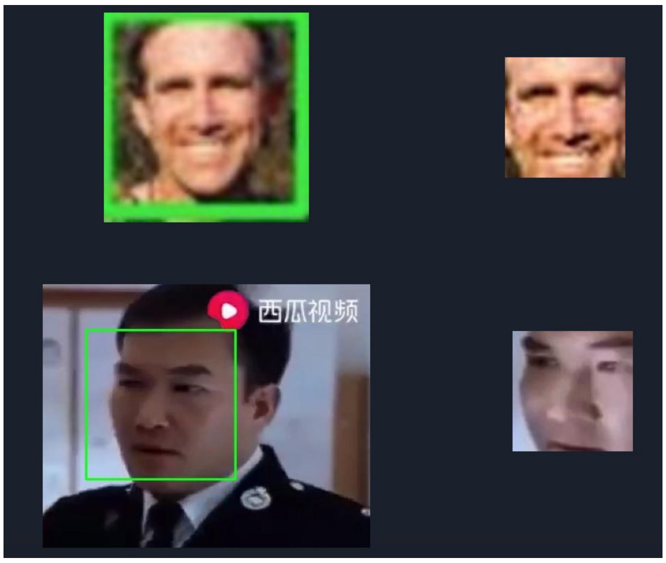
Diagram created by Yiming zhang

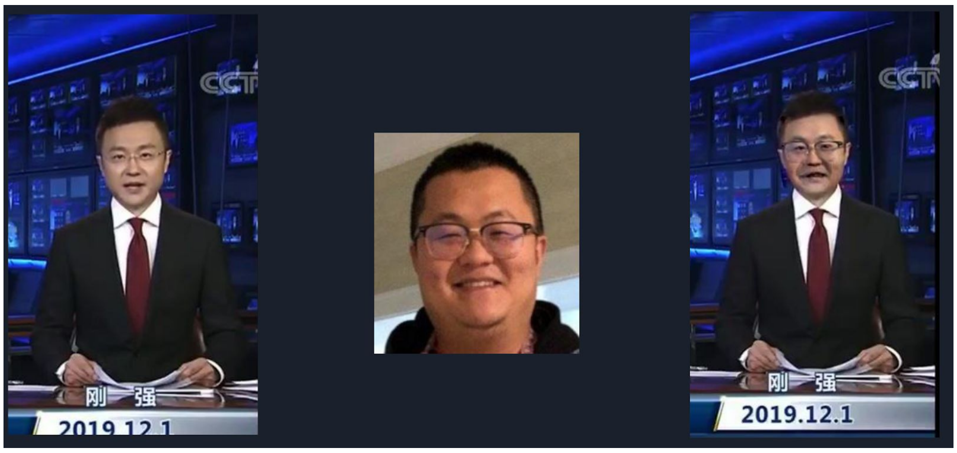
We used the weights provided by third one for demo, though it is similar to the second one, but uses less GPU memory and the training speed is faster than the second model.

3. We get the prediction of the face by model.predict_generator, then we transform the prediction to a tuple of length 3, (gender, race, age). For detection, we draw a rectangle on the bound of the face, also the text which show the information of prediction, and save this new image in the folder rect_frame.

For swapping the face, we have an input image face.jpg, also we can get the bound of the face on the frame. While if we just resize the new face to the size of the face bound and replace each value of pixels on origin position, the performance is not good, because the edges will be very obvious. Also the face bound will have other background for both old and new face.

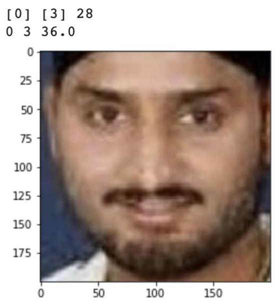
So we have ratio for width and height for both old and new face, and only replace the part within bound * ratio, for example only pick 80% of width and 90% of height of the new face, and pick 75% of width and 75% of height for old face. Then we use the Poisson Blending method to merge the edges, and the performance is much better.[5]



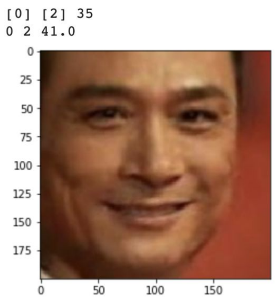


Results and discussions

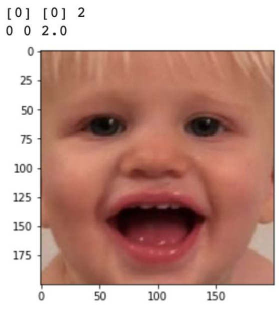
Our result is fairly good for face prediction. While it may mess up people in the dark light with black people and mess up indian with black people. And since number of asian faces in the training set is three times less than white faces, in the test phase, asian faces tend to be mess up with faces in other races, even their gender might be messed up. In a video, there is not so much frontal faces for us to detect, and most our training sets are frontal faces. So there may have changing in a the result for the same person appearing in the video.



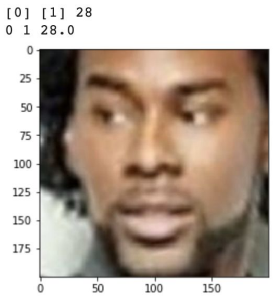
```
[0] [3] 28
0 3 36.0
```



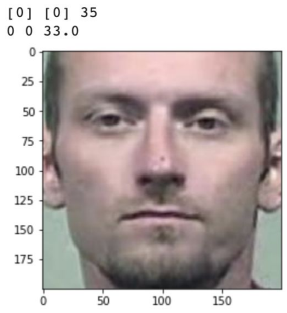
```
[0] [2] 35
0 2 41.0
```



```
[0] [0] 2
0 0 2.0
```

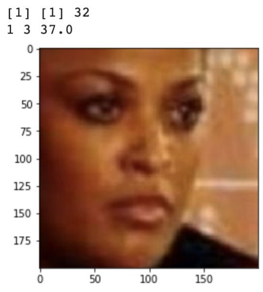


```
[0] [1] 28
0 1 28.0
```

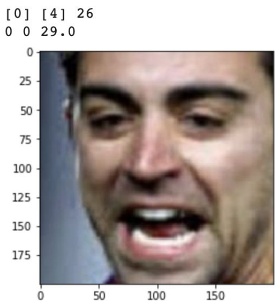


```
[0] [0] 35
0 0 33.0
```

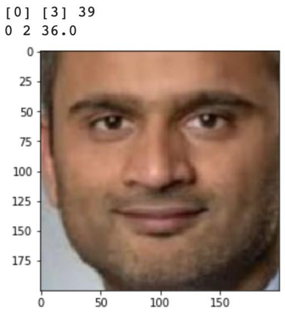
These are good prediction results, three number at the top is the label of test images and three at the bottom is our prediction, the first one is gender, the second is race and the third one age.



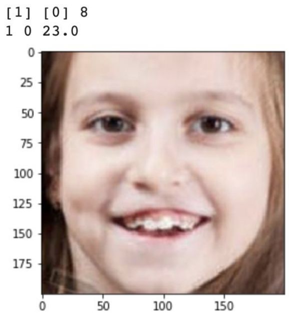
```
[1] [1] 32
1 3 37.0
```



```
[0] [4] 26
0 0 29.0
```



```
[0] [3] 39
0 2 36.0
```



```
[1] [0] 8
1 0 23.0
```

Here are some test results that have uncorrect predictions.

Main challenges

Compare to the current existing gender, race and age detection models, we are not sufficient in the precision of prediction. This is mainly because of the restriction of our network size and training size. Such kind of classification problem needs larger network to get accurate result. Since the Colab has restriction in RAM and training time, we cannot build deep network or train the data for a long time or with large dataset. Our dataset for training RGB images is not large, it only has 6000 images and among them there are many children under 3 years old, which is not likely to appear in common videos that need to be predicted. And the distribution of race is not uniform. Although we used data augmentation, we still failed to get very good trained weights.

Conclusion and future work

Our model is relatively good at predicting white people' face since their faces are a large part of our training set, but it is not good at predicting faces of asian because the lack of asian faces in the training set. Also next time we can delete a large part of baby faces and add more teenages and adults, since we seldom use video with baby faces to do face recognition or swapping.

Speaking of face swapping, our implementation is relatively rough. Since we only swap for the area that is face detected. The target of swapping is based on his/her gender, race and age. If we need to implement precise face swapping, we have to build another model to locate the eyes, nose, mouth on people's face and prepare another training set to training this model.

In the future we may get a good graphic card and train our model on our own computer, using deeper network and larger and more useful dataset.

Reference:

1. https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

2. http://vis-www.cs.umass.edu/lfw/

3. https://www.nist.gov/itl/products-and-services/color-feret-database

4. https://susanqq.github.io/UTKFace/

5. This part is from presentation ppt.