2021 Computer Vision Assignment Report.

Zhenyi Jia

May 3, 2021

1 Abstract

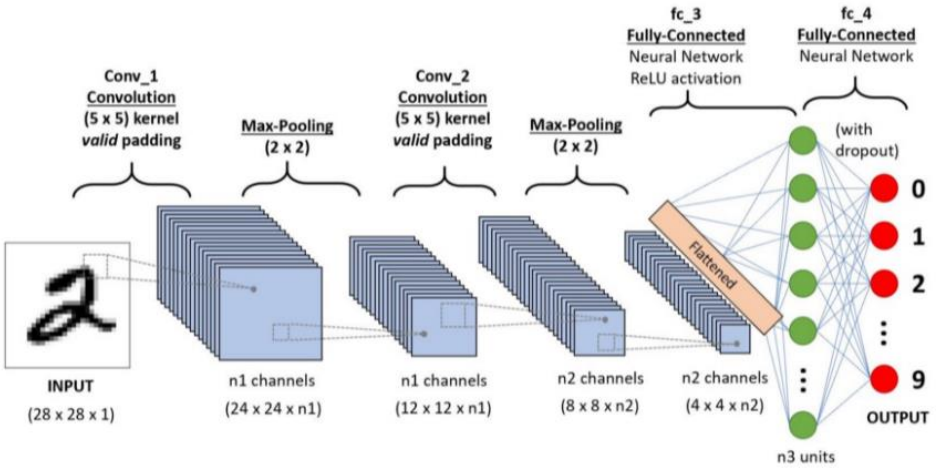
In the fields of computer vision and machine learning, detecting facial key points is a difficult and significant problem. Due to diverse influences from original photographs, very different facial features from person to person, and a lack of guidance to appropriate algorithms, capturing key points on the face is difficult. As the development of deep learning, more and more approaches focused on it are proposed for addressing related problems, ushering in a slew of ground breaking changes.

In this project, I have proposed a simplified method based on Convolution Neural Network (CNN) for solving the problem of detecting facial key points. I was provided with a list of 236×236-pixel images with their corresponding (x, y) coordinates of 46 facial key points . I train the facial key points data and assess its performance on the test set to evaluate predictions. Also, I have implemented the face segmentation based on the predicted points and added a sunglass effect on it.

2 Introduction

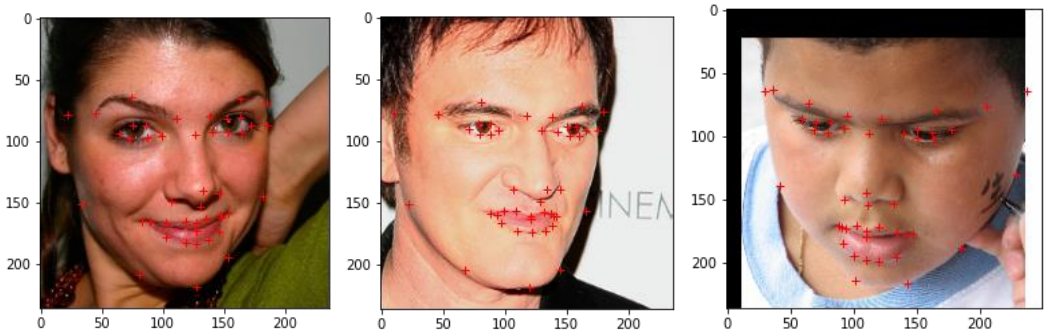
Facial key point detection is a problem of estimating the position of the eyes, nose, and mouth in a facial image and it aids in improving face recognition accuracy. Facial features may differ according to size, position, pose and expression, while image quality may vary with illumination and viewing angle.(Park, 2019)

A Convolutional Neural Network (CNN) is a Deep Learning algorithm .In simple word what CNN does is, it extract the feature of image and convert it into lower dimension without loosing its characteristics. The pre-processing required in a CNN is much lower as compared to other classification algorithms, While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.



A CNN sequence to classify handwritten digits

In this project, I have provided with the training testing and example data folders .Training data contains face images with image data as row-ordered list of pixels and the coordinates for facial key points ,Testing data and example data just have images. Training dataset consists of 2811 RGB scale training images, with 236*236 pixels each, and the (x, y) coordinate pairs of 46 features, such as eyes, nose, Face contour, etc. Following show examples of the face image with 46 key points marked by red plus.

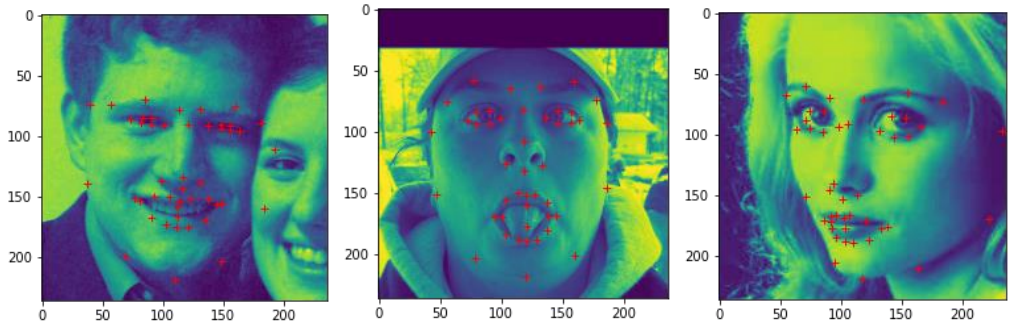


Random Training Dataset with key points

3 Methods

3.1 Data processing

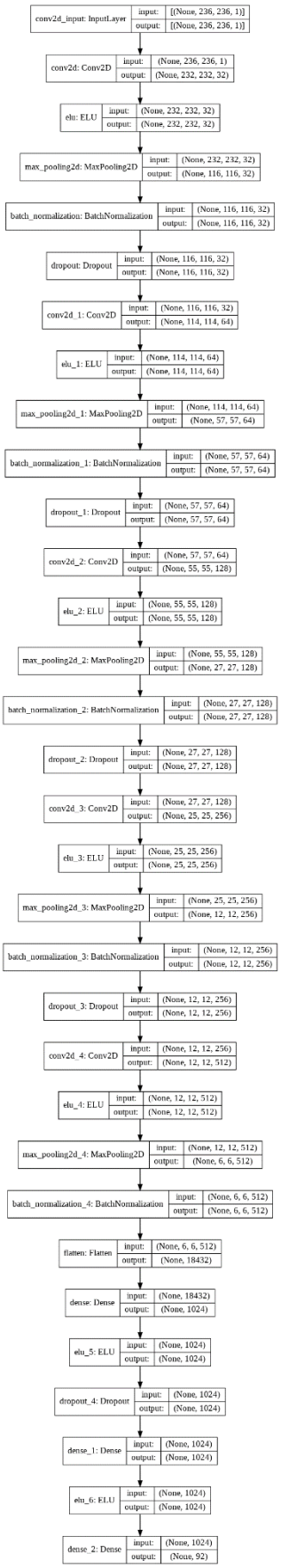
Input layer in CNN should contain image data, train data in this project represented by three dimensional matrix as I show it earlier. And it is necessary to reshape train data into a single column.



Random Grayscale Training Dataset with key points

Besides that ,train points also need to normalized. I reshape it from (2811, 46, 2) to (2811, 92) . the output class should be 92 because 92/2 = 46 and the predict points should be 46 key points. Now I have my processed data, next I can train it using deep learning frameworks – CNN.

3.2 Convolutional Neural Network



The brief architecture diagram of My CNN is shown in Figure 1 and the full diagram is shown at left.

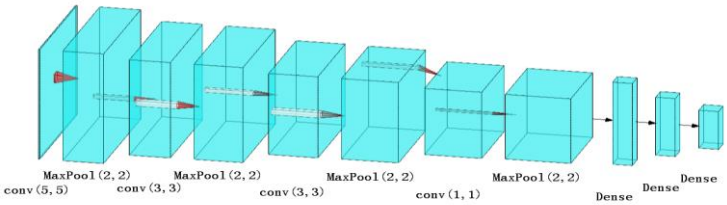


Figure 1 my CNN

In CNN, I have applied four convolutional layers. The first step is to create convolution layer 1 in our network. The first element in the layer definition is Conv2d which creates a set of convolutional filters. The kernel size argument is the size of the convolutional filter, in this case I choose 5 x 5 sized convolutional filters. Then I have applied ELU activation to this layer. The reason I use ELU is it combine the good parts of ReLU activation and leaky ReLU activation. The last element that is added in the layer definition is the Maxpooling operation. We then use batch normalization to limit covariate shift by normalizing the activations of each layer which transforming the inputs to be mean 0 and unit variance. This allows each layer to learn on a more stable distribution of inputs, and would thus accelerate the training of the network. I also specify a drop-out layer to avoid over-fitting in the model.

The second, third and fourth convolution layer are defined in the same way as the first layer. The only difference is the values for input channels and output channels depending on each layer.

The final output layers are fully connected (FC) layer where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network. I also use drop layer after one times FC layer.

During the compile ,I have selected Adam to be our optimization algorithm since it is best suited for our deep learning frameworks and loss function is MSE (Mean Squared Error).

3.3 Face Alignment

Key idea of face alignment is Affine transformations. The align function extracts the location of the eyes in the original face, calculates the centre of each eye, calculates the angle between the eye centre line and the horizontal line, and then calculates the left and right eye coordinates in the aligned target image. Then, in the original image, measure the centre coordinates of the line connecting the two eyes, use this as the rotation centre, and angle as the rotation angle to rotate the image by Affine transformations. Resulting in a scaled image finally. The key points array also need to be rotated.

3.4 Face segmentation and graphical effects

The simple method to implement the face segmentation through facial key points is use fillpoly method. In first time I tried to use points directly and it becomes a failure cases(figure 2), then I found the correct way is to just use contour key points.

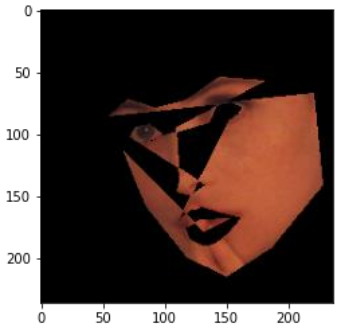


Figure 2 failure case

START

Mask=Make a mask with all black from image;

points =Get face contour key points;

cv2.fillPoly(mask, np.int32([points]), wanted colour);

cv2.bitwise_and(np.uint8(image), mask);

STOP

pseudocode with face segmentation

At graphical effects part, I tried to add red lips on mouth and sunglass image based on eye key points.

4 Results

Except the final CNN model which I mentioned before, I also had a failed attempt. I plot the graph of training and validation model summary for them. Figure 3 is the graph of loss value and accuracy for the failed model through epochs, the next figure is for the final model(Figure 4). It shows directly that failed model is over-fitted so I use ELU activation to replace ReLU activation and change kernel size of each layer. The final results looks better than before.

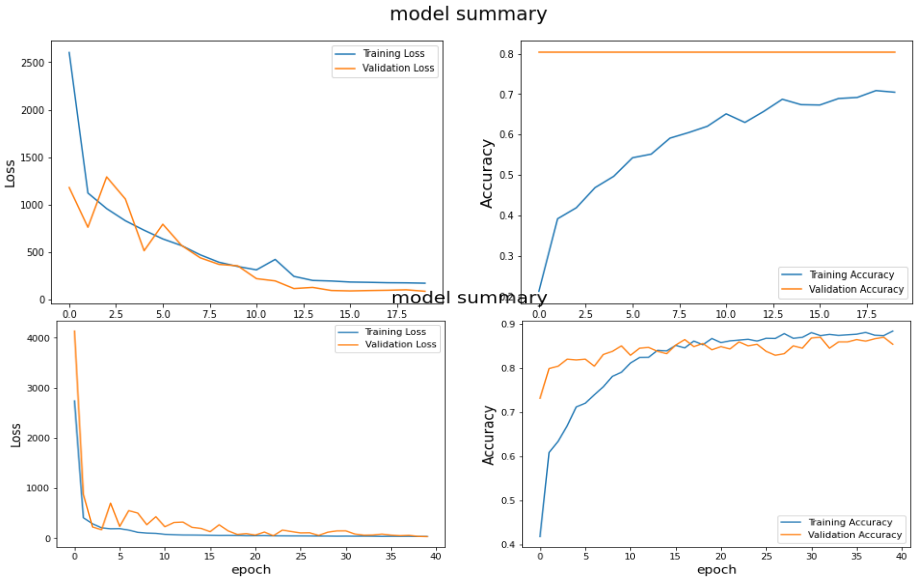


Figure3. failed model summary

Figure4. final model summary

I visualized the results of test image and get some good predictions and bad predictions at Figure 5.

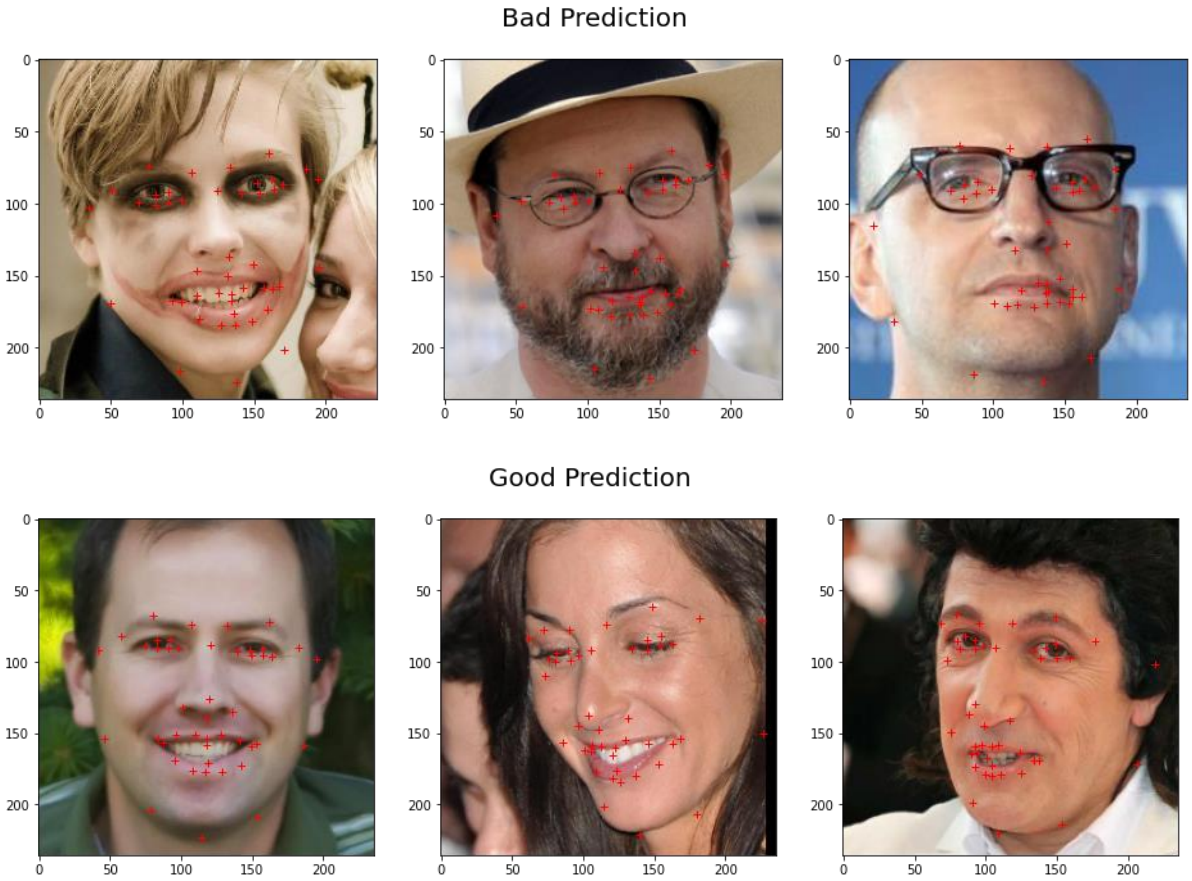


Figure 5 good predictions and bad predictions

Figure 6 shows The results of face alignment, face segmentation and graphical effects systems at Qualitative examples.

My CNN model can predict most key points of images such as the location of eyes, face contour and eyebrow. But it is helpless to some difficult cases such as the fourth case with open mouth, it is hard to detect the non-closed mouth and point the wrong location to mouth and nose.

At the same time , I also meet problem with facial effects, it is hard to locate the correct location of sunglasses. It seems is just a simple attempt at the scope of my ability and only works well for standard frontal image like the second case.

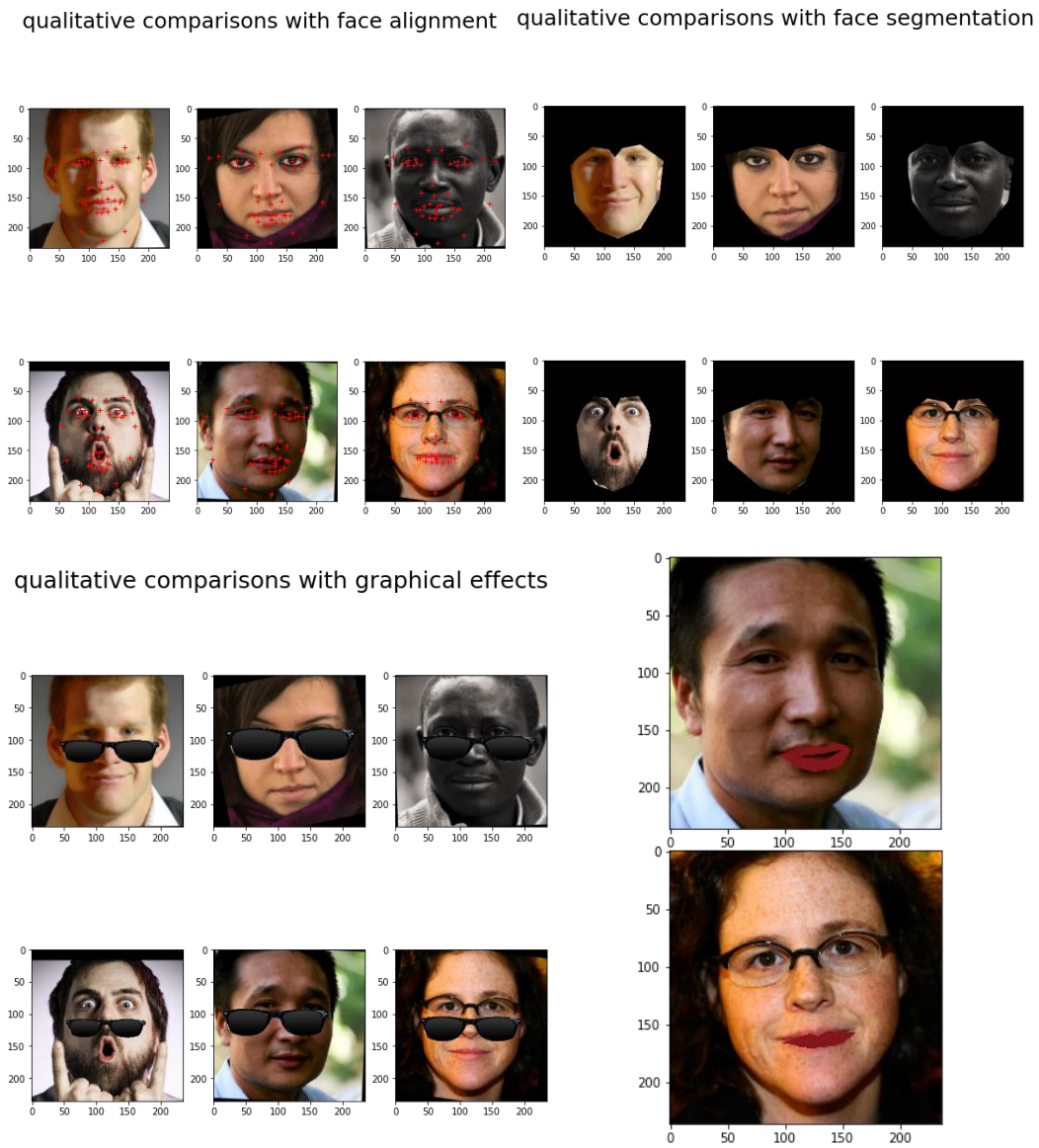


Figure 6 Qualitative examples

5 Discussion

I have envisaged the possible ideas of improving performance of CNN.

Provide a larger input size use the image augmentation, expanding the training datasets and add extra layers, which can lead to better results, but that limited by GPU performance.

CNN can extract features automatically , do more artificial feature extraction could help CNN to improve the correct rate of detecting mouth.

6 References

[1]Park, J.K. and Kang, D.J., 2019. Unified convolutional neural network for direct facial keypoints detection. *The Visual Computer*, *35*(11), pp.1615-1626.

[2] Zhang, S. and Meng, C., 2016. Facial keypoints detection using neural network. *Stanford Report*, *1655*, p.1655.

[3] Y. Sun, X. Wang, and X. Tang. Deep convolutionalnetwork cascade for facial point detection. InPro-ceedings of the IEEE Conference on Computer Visionand Pattern Recognition, pages 3476–3483, 2013.

[4] Q. Wang and J. Yang (2006). Eye Location and Eye State Detection inFacial Images with Unconstrained Background. Journal of Informationand Computing Science, vol. 1(5), pp. 284-289

[5] A. Ng (2014). Lecture Notes for CS229 - Machine Learning. StanfordUniversity.