Computer Vision Assignment Report

By 246743

Abstract

Face alignment is one of the main steps in popular face recognition techniques nowadays, and a common way to train a model to predict a face is through a dataset that has been labelled. In this report, three different Convolutional Neural Network (CNN) models will be trained by using TensorFlow [1], with different layers and different processing content. The test set (and also the validation set) will be applied to all models to see and analyse their euclidean distances. A series of prediction points will be printed out to give the most intuitive feedback. The three models will be compared with each other to analyse how different operations, or the number of layers, might affect the results. More specifically, my final data achieves an accuracy of around 93% in the training set and 86%-88% in the validation set, with a mean euclidean dist of between 4.4 and 4.6. In addition, lip and eye colour conversion has been achieved.

Introduction

For face alignment, there are generally two models available, the MLP model and the CNN model. The reason for not using the MLP model here is that it is difficult to train and easy to overfit due to the large number of MLP parameters. Also, MLP is less able to capture the local feature structure in the input feature map.[2]

Therefore, I chose a CNN as the vehicle for the experimental model, which is characterised by its ability to accept matrix inputs and to introduce operations such as pooling and DropOut to prevent overfitting.

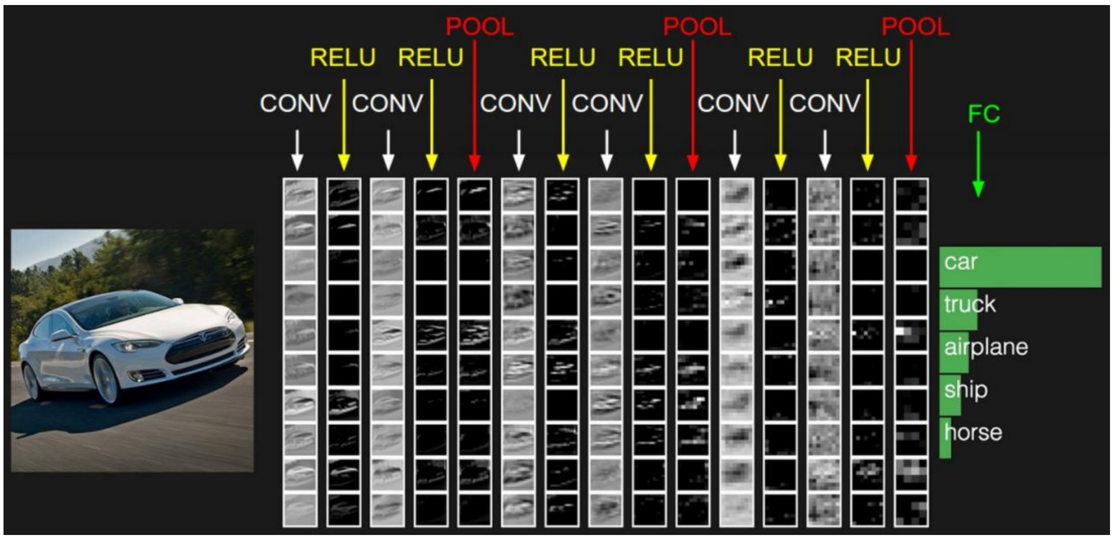


Fig 1

As shown in the Fig 1, convolutional neural networks usually contain the following layers: Convolutional Layer, Activation Layer, Pooling Layer and Fully Connected Layer(Dense Layer)

1. **Convolutional Layer** [3]

   The layer is made up of individual network nodes, and the back-propagation algorithm is applied to the transfer parameters on the network to make it more accurate and to keep progressing at all times.

2. **Activation Layer** [4]

   In fact, it is strictly speaking just a parameter of the convolution layer. Non-linear functions are often used as excitation functions embedded in the convolutional layers. The non-linearity is equivalent to transforming the space, and when the transformation is complete, it is equivalent to simplifying the problem space, so that problems that were linearly unsolvable now become solvable. This is one of the reasons why deep learning is so effective.

   In the image above, the RELU function is used [5], so it can also be called the Rectified Linear Units layer (ReLU layer).

3. **Pooling Layer** [6]

   By dividing the input image into rectangular regions and outputting a maximum (maximum pooling) for each sub-region. This is because the relative position of a feature on the whole image is more important than its exact position. The image can be successively pooled and then continually reduced, which reduces the number of parameters and the amount of computation. It also suppresses the occurrence of some over-fitting.

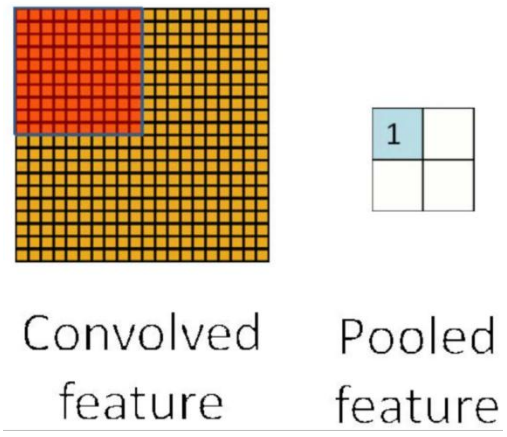


Fig 2

4. **Fully Connected Layer** [7]

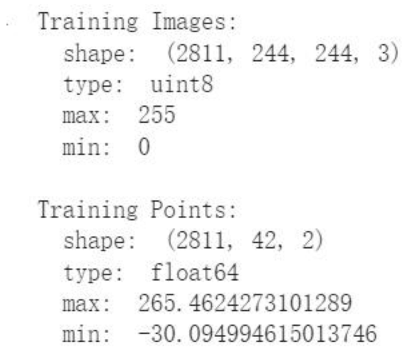
   This layer is fully connected to all nodes in the previous layer. Its role is to integrate the local feature information from the previous layer.

Methods

The procedure of experiment includes pre-processing, defining the model, training , prediction and analysing data.

1. Pre-process

   First I printed out some basic data of training data and points.



```
Training Images:
  shape:  (2811, 244, 244, 3)
  type:  uint8
  max:  255
  min:  0

Training Points:
  shape:  (2811, 42, 2)
  type:  float64
  max:  265.4624273101289
  min:  -30.094994615013746
```

Fig 3

   As can be seen in Fig 3, there are 2811 training data, where the maximum value of the pictures is 255 and the minimum value is 0, while the maximum and minimum values of the points are around 265 and -30.

The first thing is normalisation. I standardised the dtype of the images and points (float32). For the point data I borrowed the make_pipeline function and the MinMaxScalar tool to reduce the data to the interval -1 and 1, according to their respective shapes.

Then came the extraction of the validation set. To calculate the Euclidean distance, about 20% of the data was extracted manually as the validation set.

2. Define the model

In this report I have created a total of three models for comparison, the design aims of which are largely in line with the flow in the diagram below (Fig 4). The difference between the models lies in the number of layers and whether there are steps to deal with over-fitting.

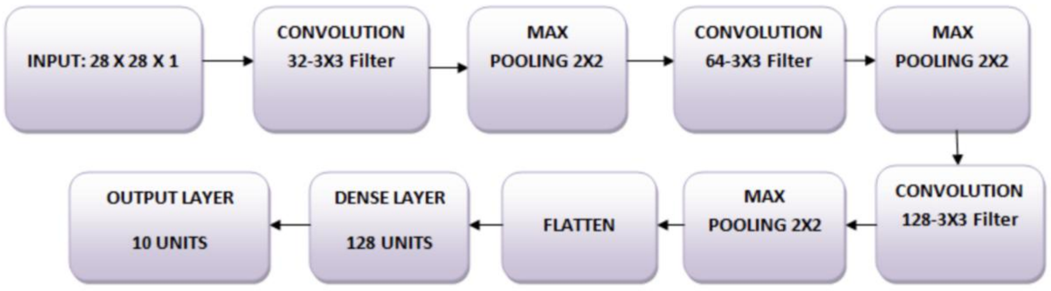


Fig 4

The first model is a CNN model with five layers, including three convolutional layers and two fully connected layers. The BatchNormalization tool is used at the beginning of the model to perform further normalisation operations. Each convolutional layer contains RELU as the excitation function and a maximum pooling layer to shrink the data. For the fully-connected layer, it is finally output with unit=84, as our points data were already reshaped to (2811, 84) in the previous pre-processing.

For the choice of kernel_initializer, 'he_normal' was used. It is a more adaptive initialisation method derived from ReLU/PReLU, which can effectively converge to about 30 levels with less error. (Fig 5)

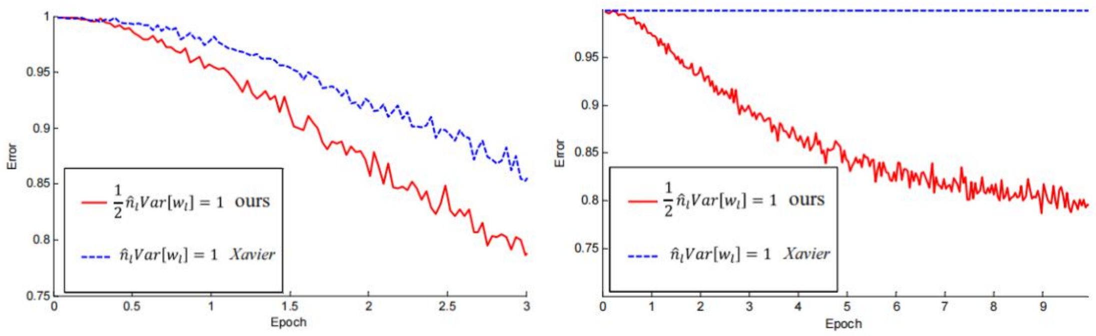


Fig 5

The second model is basically the same as the first in terms of parameter assignment and selection, the only difference being that I have increased the number of convolutional layers to 8 to see if the number of layers has any effect on the output. The third model, again, inherits the second model and adds a DropOut layer to further prevent overfitting.

3. Training

Adam is used for optimizer, with a learning rate of 0.001 as Kingma (2014) [9] states that it can achieve lower loss values than a range of other optimisers such as SGD.The collaboration between Adam and DropOut also produces excellent work for less training cost. (Fig 6)

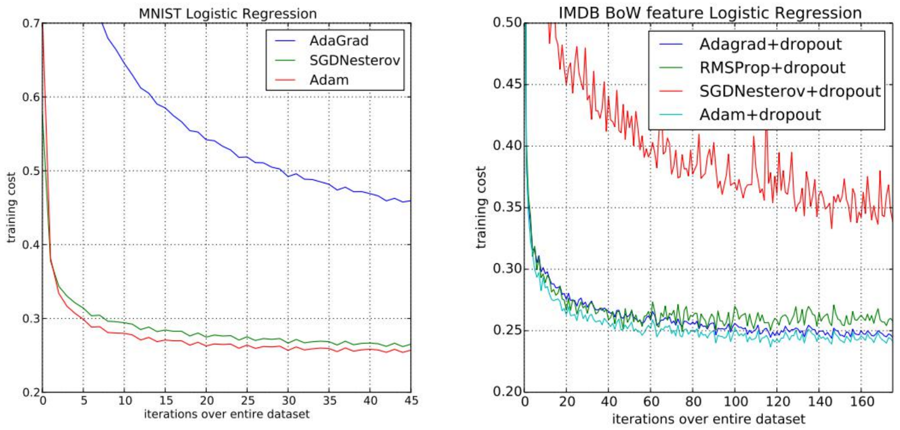


Fig 6

For the Loss and Metrics, 'mse' and 'accuracy' has been used.

4. Prediction

Predictions were made for test images and example images and the images were output. The points predicted by the model need to be formatted back (-1, 42, 2) before prediction. This is so that the images will fit when displayed.

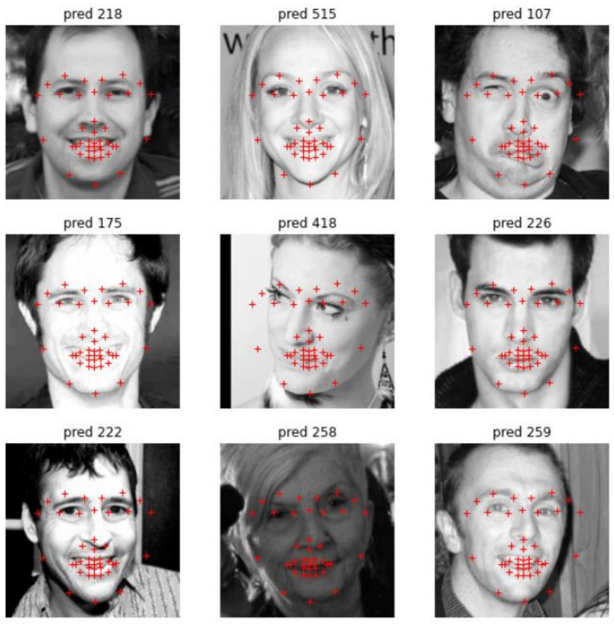
A random print function is defined to output the images as a group, which makes it easier to see the images.

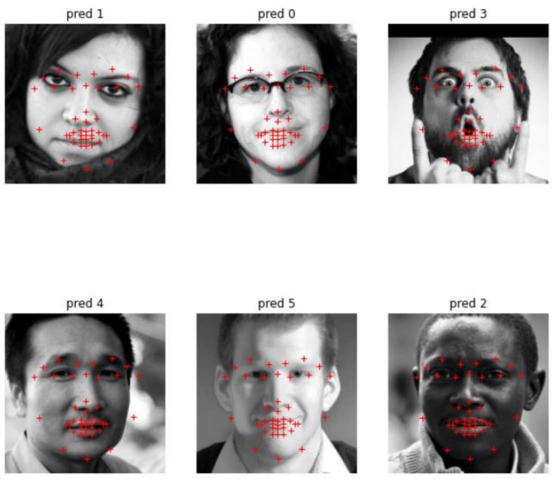
5. Analysing data

The trained model is used to predict the validation set, and the results are compared with the ground truth of the validation set to determine how well the predicted data fits the ground truth by calculating the average of Euclidean Dist.

Results

For the **First** model, the prediction of test set and examples are:

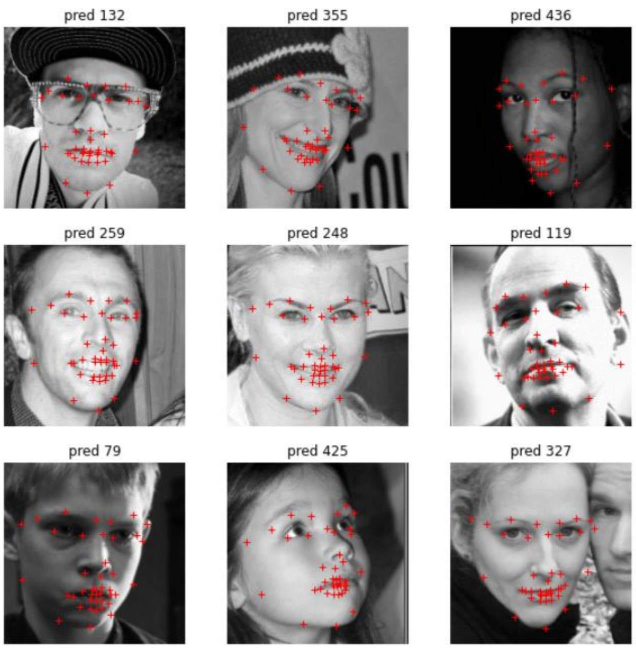


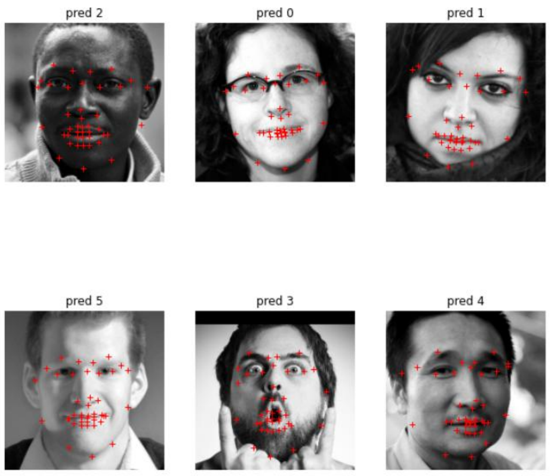


```
Mean of euclid dist of model 1 is  11.92315
```

Fig 7

For the **Second** model, the prediction of test set and examples are:

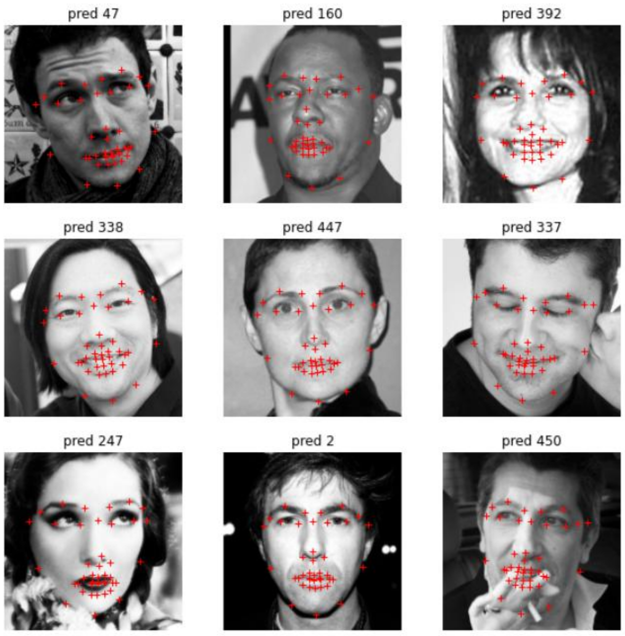


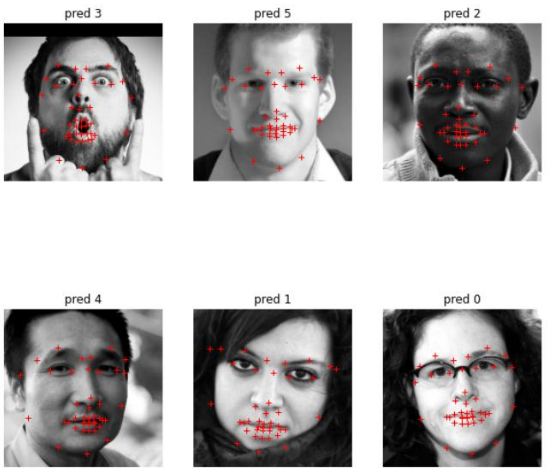


```
Mean of euclid dist of model 2 is  4.6931667
```

Fig 8

For the **Third** model, the prediction of test set and examples are:





```
Mean of euclid dist of model 3 is  4.575579
```

Fig 9

Discussion

The predictions of the first model are far worse than the latter two, as model 1 has only 5 layers, while both model 2 and model 3 have 8 layers.

In the predictions of model 1, the faces are roughly present. For example, in prediction 418 in Fig 7, the general orientation of the entire face is formed, but the fit to the face is not as good as it could be, causing the model to treat the background on the left side of the picture as part of the face as well, and something similar happens in prediction 259.

In models 2 and 3, the predictions were much more accurate, with most of the predicted points being on the boundary lines of the facial contours. Model 3 adds a DropOut operation to model 2, as can be seen from the euclid dist of both. The scores for model 3 are always slightly smaller than those for model 2, implying that overfitting prevention is effective.

Failure Cases

There is no treatment of possible outliers in the data during the pre-processing phase. If NaN is removed or replaced by averages, the trained model will be theoretically better.

Another failure case is the first model. Fig 10 is a comparison of the prediction points between the different models for the same image, and you can see that model 1 misjudges the hair as part of the face. The other two models are more accurate.

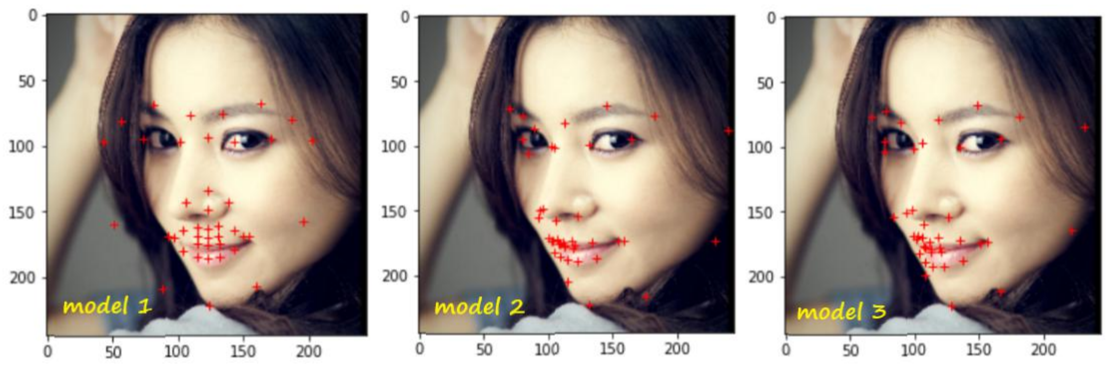


Fig 10

Extension: modify the color of lips/eyes

Result:

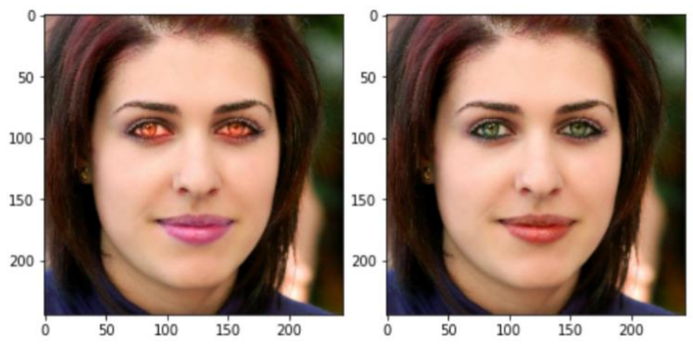


Fig 11.1, color modification on training set

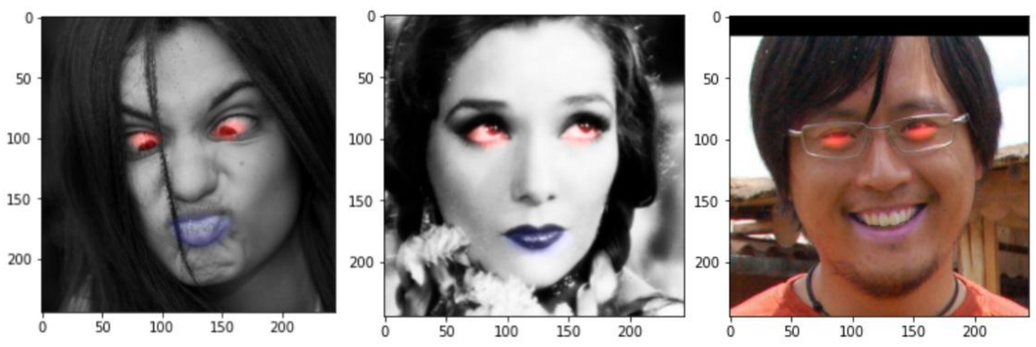


Fig 11.2, color modification on testing set

Handling of Eye Coordinate Points:

As shown in Fig 12, there are only 2 dots for each eye and they only form a line. My solution is to add two more points for the top and bottom of each eye. Their coordinates are derived from the original coordinates by calculation.

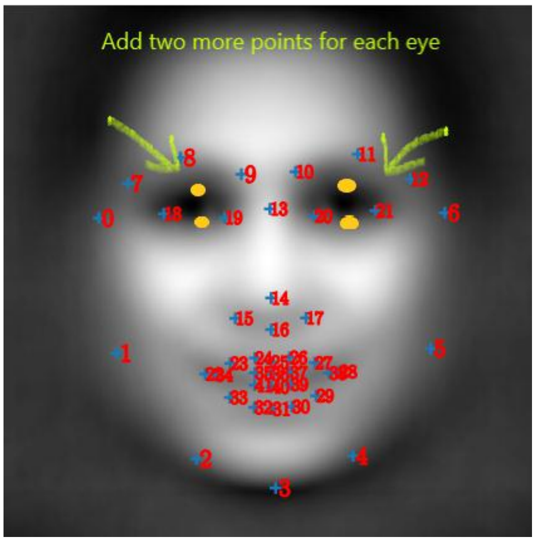


Fig 12

Process of making a mask:

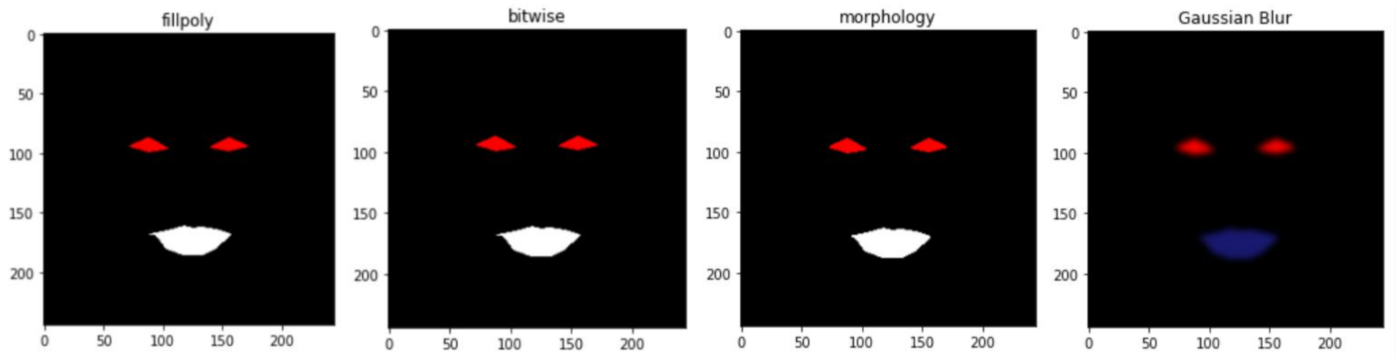


Fig 13

The sequence of making a mask is shown in Fig 13, the morphology include both erosion and dilation. In actual results the modification of the lips' colour is generally better than that of the eyes, since the eye has only four characteristic points, considerably fewer than those for the lips.

Conclusion

In general, I have experimented and analysed using three different models by comparing them, and the best model has a

Mean of Euclidean Distance that is usually between 4.2 and 4.6, which is a satisfactory result. If we want to improve this, we need to pre-process the training data better and try more combinations of parameters and more epochs.

For the colour modification, I did a mask for both eyes and lips, and then combine the mask and the original image to make the image "change the colour". The result has a certain amount of success, but is limited by the accuracy of the prediction points, so it is not possible to achieve a perfect visual match to the testing image.

References

1. Ballard, W. (2018). *Hands-On Deep Learning for Images with TensorFlow: Build Intelligent Computer Vision Applications Using TensorFlow and Keras*. Birmingham: Packt Publishing, Limited. https://ebookcentral.proquest.com/lib/suss/reader.action?docID=5485032
2. Guo, M.-H., Liu, Z.-N., Mu, T.-J., Liang, D., Martin, R. R., & Hu, S.-M. (2021). *Can Attention Enable MLPs To Catch Up With CNNs?* https://arxiv-org.ezproxy.sussex.ac.uk/abs/2105.15078
3. Tensorflow creator (2022) *tf.keras.layers.Conv2D* https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
4. Tensorflow creator (2022) *tf.keras.layers.Activation* https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
5. Schmidt-Hieber, J. (2020). *Nonparametric regression using deep neural networks with ReLU activation function.* The Annals of Statistics, 48(4). https://doi.org/10.1214/19-AOS1875
6. Tensorflow creator (2022) *tf.keras.layers.MaxPool2D* https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
7. Tensorflow creator (2022) *tf.keras.layers.Dense* https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* 2015 IEEE International Conference on Computer Vision (ICCV), 1026–1034. IEEE. https://doi.org/10.1109/ICCV.2015.123
9. Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization.* https://arxiv-org.ezproxy.sussex.ac.uk/abs/1412.6980