

Computer Vision Report

By 184510



29 May 2020

University of Sussex

Computer Vision

# Introduction:

In this report I will go through how I have created two models that perform face alignment on the testing data provided. First one is just a multi-layer feed forward network model and second one is using a convolutional neural network. I will also go through how both models perform, talk about their strengths and weaknesses and compare them to each other. Further in the report I will also talk about how I have implemented the face segmentation and how I have added the simple graphical effect.

# First Model:

## Brief introduction:

My first model is using a multi-layer feed forward network [Figure 1]. MLFFN “is an interconnection of perceptrons in which data and calculations flow in a single direction”[1]. As show on the diagram [Figure 1] each node from the input layer (Dense 1) is connected to each node within the hidden layer (Activation 1) and the data from the input layer is flowing forward to the hidden layer. Then the nodes in hidden layer are connected to all the nodes within the output layer (Dense 2). The number of layers within the network is the amount of perceptrons.

## Methods I have used:

For the first model I have started by doing pre processing on the training and testing data. I have stored the images and points of the training data and images of the testing data into variables. Then I have turned all the image into grey scale and all three variables have been set to float32 format. In order to set images into grey scale I have used “skimage” library [4]. Then I made sure that all the coordinates of the points (0-250), points (-1, 1) and images (0, 1) are within the correct range respectively and that they are all in the format that my model can use.

Next, I have created the first model and set it to the sequential object from “Keras” library. Then I have applied the dense layer to the model that also acts as the input layer [2][Figure 2]. Then the activation layer applies the activation function [3][5] that does not change the output shape [Figure 2 and 3]. Then I applied dense layer that was set to 136 which represents the coordinates for all the points, by this output shape changed to 136. This was also an output layer [Figure 1, 2 and 3] . Then I defined an SGD which was used as the optimizer when compiling the model after that I trained the model. I have also written a function that plots the points on to the images.

# Second Model:

## Brief Introduction:

My second model is using a convolutional neural network [Figure 4]. CNN is a deep learning algorithm that takes in an image as an input and assigns importance such as weights and biases to different aspects and object in the image and be able to tell which one is which[7]. As shown on the diagram [Figure 4] the data in this model is only flowing forward.

## Methods I have used:

For the second model I have applied the same pre-processing as for the first model however the reshaping was different. For the training and testing images I had to reshape them into a 4-dimensional array. I have set the first dimension to be the total number of the images, then the next two dimensions specify the size of the images and the last number specifies that the images are in black and white. Then for the training points I have set the first dimension to the number of photos with points and second dimension was the number of coordinates which is double the amount of points on every image.

Next, I have created the second model and set it to the sequential object from “Keras” library. Then I have added a 2d convolutional layer that calculates convolutions in batches. Then I have applied the activation layer that haven’t changed the output [Figure 4]. Then I have added a max pooling layer [Figure 4 and 5]. The activation layer makes the size to be 4 times smaller than previously and does this by taking the maximum value from each region which reduces the dimensionality of our feature map [8][Figure 5]. Then the 2d convolutional layer is performed again and doubles the amount of the feature maps. Then the processes repeat as shown on the diagram [Figure 4] until there is 128 feature maps and each map are of size 30 by 30. Then the flattening is performed on the output so the 4-dimensional array is changed to a single dimension array that can be inputted into the multi-layer feed forward network [Figure 4]. Then the dense and activation is performed on the model alternately until the output shape was 136. Then I defined an SGD which was used as the optimizer when compiling the model after that I trained the model. Then I have used the plot function to plot the points onto the images. During this model I have decided to train both models on 250 epochs as the results have been very pleasing and the compiling time wasn’t extremely long.

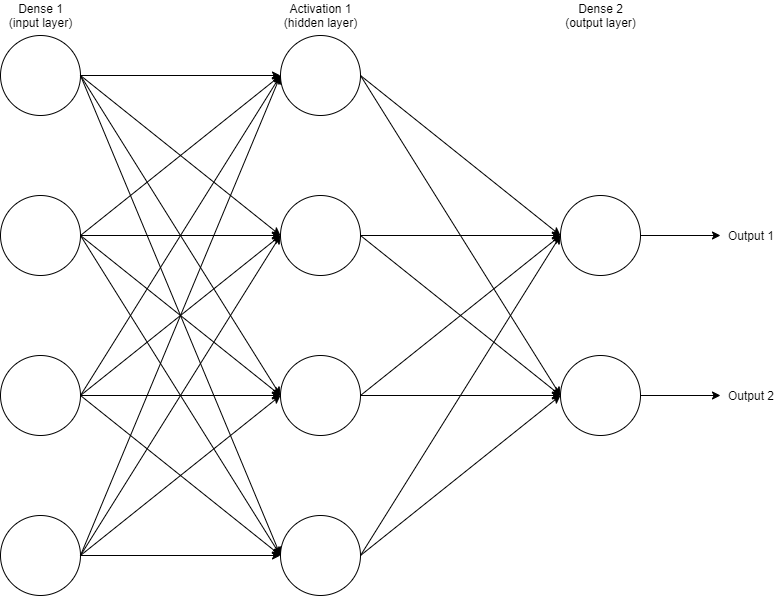


Figure 1 - Diagram representing multilayer feed forward network



Figure 2 - Flowchart representing model 1

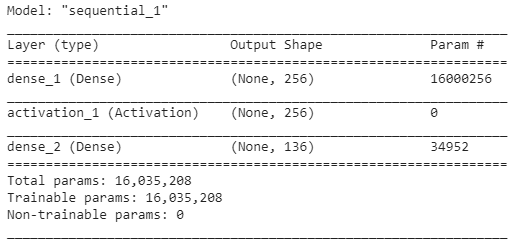


Figure 3 first model summary

Obraz zawierający rysunek

Opis wygenerowany automatycznieObraz zawierający rysunek

Opis wygenerowany automatycznie

Obraz zawierający rysunek

Opis wygenerowany automatycznie

Figure 4 Diagram representing a convolutional neural network

Obraz zawierający zegar

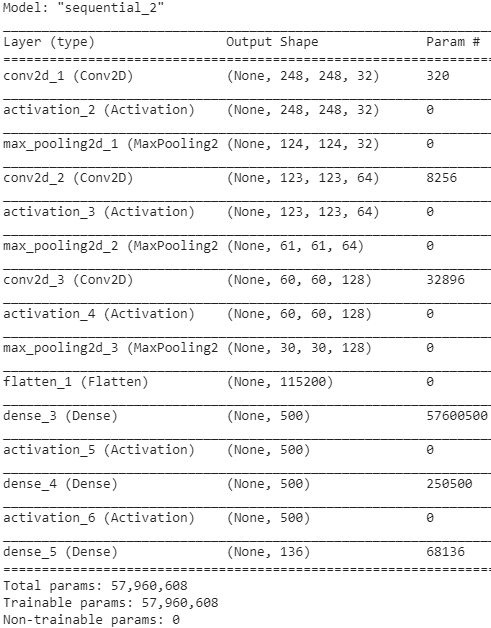
Opis wygenerowany automatycznie

Figure 5 the convolution operation

Figure 6 second model summary

# Comparison of the models:

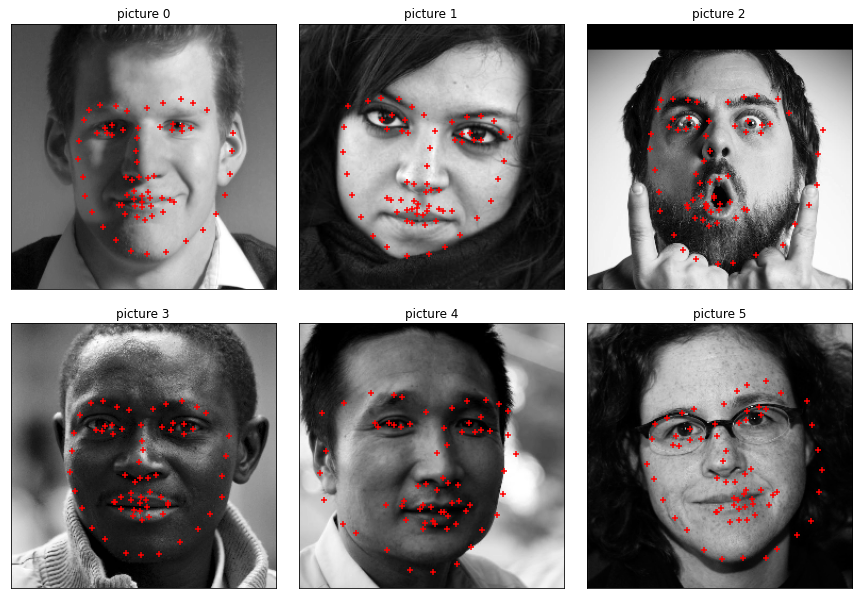


Figure 7 Model one point prediction

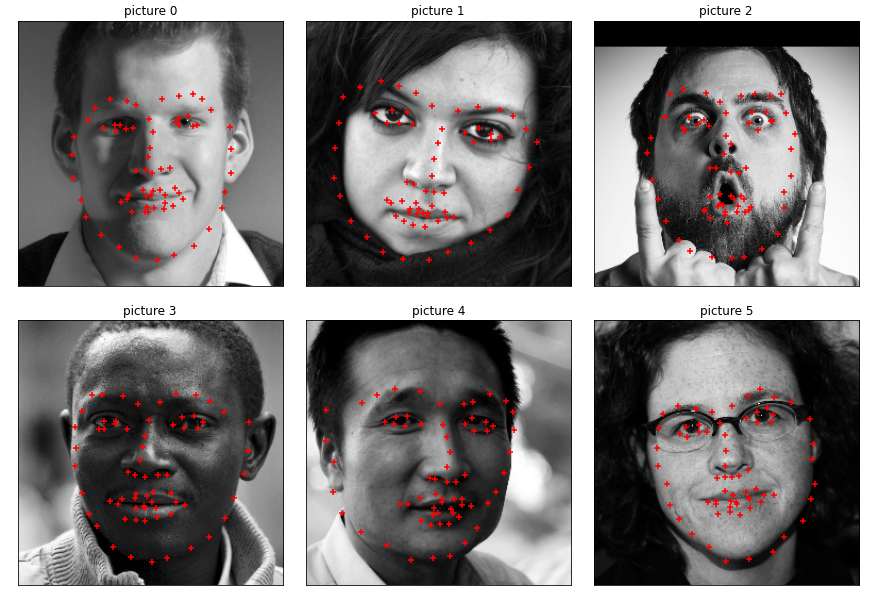


Figure 8 Model two point prediction



Figure 9 Euclidean Distance model one



Figure 10 Euclidean Distance model two

Looking at the photos and the Euclidean distance comparison it is very clear that the second model is superior to the first one [Figure 7, 8, 9 and 10]. Looking at the “Picture 1” the second model was way better at plotting the points of the face shape as well as points for eyes, mouth and nose. “Picture 3” is also a quite interesting example as even that both models are quite good at plotting the points on the face, the second model is still way more accurate compared to the first one when looking at details such as mouth. “Picture 2 and 4” show that even my second model could use some improvement as it’s still struggling with certain photos. “Photo 2” caused trouble to my model as the face expression on it is quite unique and uncommon and this could have affected the performance of the second model. Looking at all photos with points plotted by the second model [Figure 8], the biggest area of improvement is plotting the points for the mouth on the photos.

# Failure Cases:

The first failure case was a model that was calculating an average position of each point. This model was a very basic model that would put the points in the same position for each photo. The second failure case I had was when I tried to write a third model that was going to improve my model even further. The idea was for each photo to be shifted slightly multiple times. This would theoretically improve the performance of the model however it would make it much more complex which I decided was unnecessary. The complexity that this would bring to the model was simply not worth it as the gain in performance would not justify it. The third and last failure case was when I tried to use “DLIB” library. However, after reading the specification carefully I realised that we are not allowed to use that library as it solves the problem for us.

# Face Segmentation:

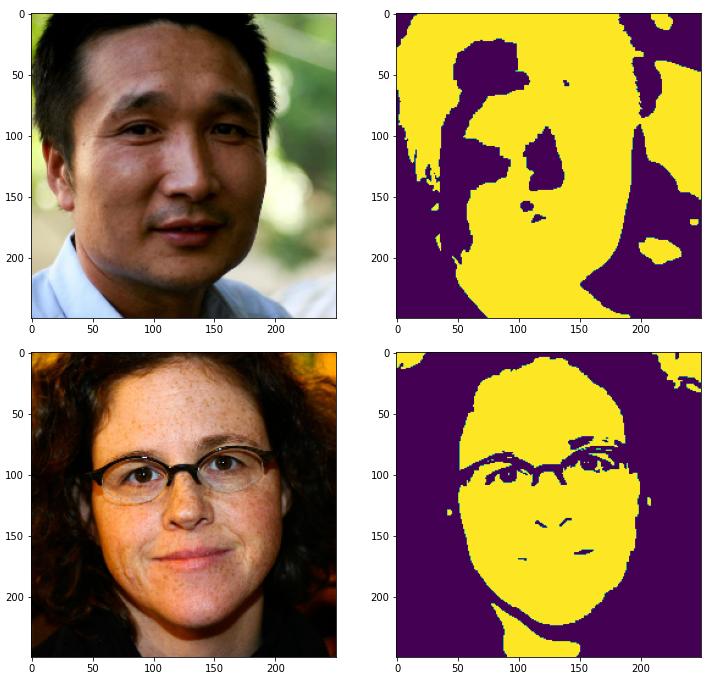
 

Figure 11 example photos before and after face segmentation

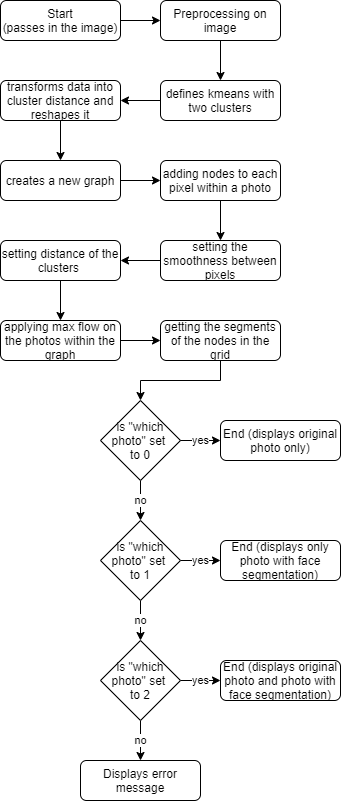
For the face segmentation task I have decided to use the graph cut segmentation approach. The process of how I have designed my system has been represented by the flowchart [Figure 12]. My face segmentation system performs quite well as seen on photos 4 and 6 [Figure 11]. The only time that the system struggles is when the face has a shadow on it. As my model only uses two colours and the shadow changes the contrast drastically the face segmentation is not able to perform correctly as seen on photos 1 and 5 [Figure 11].

Figure 12 flowchart for face segmentation process

# Graphical Effect:

For the graphical effect, I have decided to add “Harry Potter” scar to the photos [Figure 13]. The process has been represented with the flowchart [Figure 14]. As seen on the photos the effect is fairly pleasing and the only photo that is having issues is the second photo as the scar should be under the hair. In order to fix this, I would require points to keep track of the hair line.

Figure 13 graphical effect added on the example images

Obraz zawierający pomiar, znak, parking, ulica

Opis wygenerowany automatycznie

Figure 14 flowchart for graphical effect process

# Summary:

Overall, I have been very pleased with my achievements. The only thing I would improve if I could, would be the performance of my face alignment system. Even that my model performed well there is still room for improvement that could make the euclidean distance value to be even lower.

# References:

1. Unknown (2007) *Multilayer Feedforward Neural Networks.* Available at: <https://docs.roguewave.com/imsl/c/6.0/stat/default.htm?turl=multilayerfeedforwardneuralnetworks.htm> (Accessed: 20/05/2020)
2. Developers of tenserflow (2020) *tf.keras.layers.Dense.* Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense> (Accessed: 20/05/2020)
3. Developers of tenserflow (2020) *tf.keras.layers.Activation.* Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation> (Accessed: 20/05/2020)
4. Developers of skimage (unknown) *RGB to grayscale.* Available at: <https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_gray.html> (Accessed: 20/05/2020)
5. Brownlee J. (2019) *A Gentle Introduction to the Rectified Linear Unit (ReLu).* Available at: <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/> (Accessed: 20/05/2020)
6. Developers of tenserflow (2020) *tf.keras.layers.Conv2D.* Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D> (Accessed: 21/05/2020)
7. Saha S. (2018) *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way.* Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (Accessed: 21/05/2020)
8. Ujjwalkarn (2016) *An Intuitive Explanation of Convolutional Neural Networks.* Available at: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/> (Accessed: 21/05/2020)
9. Developers of tenserflow (2020) *tf.compat.v1.layers.MaxPooling2D.* Available at: <https://www.tensorflow.org/api_docs/python/tf/compat/v1/layers/MaxPooling2D> (Accessed: 21/05/2020)
10. Developers of maxflow (unknown) *Tutorial.* Available at: <http://pmneila.github.io/PyMaxflow/tutorial.html> (Accessed: 21/05/2020)