Report

Facial Detection

A picture containing text, athletic game, sport

Description automatically generated

Cristian Sirbu

Cand no: 230982

Table of Contents

[Overview 3](#_Toc103216275)

[Introduction 3](#_Toc103216276)

[How They Work 4](#_Toc103216277)

[Model 6](#_Toc103216278)

[Performance 7](#_Toc103216279)

[Additional Task 8](#_Toc103216280)

[Conclusion 8](#_Toc103216281)

[References 9](#_Toc103216282)

# Overview

Face recognition has gotten a lot of attention in recent years, and its research has been rapidly expanded by not only engineers but also neuroscientists because it has a lot of potential applications in computer vision communication and automatic access control systems. Face detection is an important aspect of face recognition because it is the initial stage in automatic face identification. However, face detection is difficult because there are numerous image appearance differences, such as position variation (front, non-front), occlusion, image orientation, illumination condition, and facial expression.

# Introduction

In this paper, I created, built, and tested a system for face alignment, also known as finding facial landmarks in photos. I have got three sets of photographs for this report (training images, test images, and a few samples). The train set includes 2811 photos with 255X255 resolution and three colour channels (RGB), as well as 42 face key points and their x and y coordinates. I will test my prediction on 554 photographs and evaluate the performance of my module on the example set. I will train a convolutional neural network on labelled photos and predict facial key points on test cases with high accuracy. In addition, as part of this report, I will create a function that changes the colour of image portions that correspond to predicted key points. I will utilise Python libraries like OpenCV for image editing, Matplotlib for visualization, and Keras for CNN to put it into practice.

Graphical user interface, application

Description automatically generated Graphical user interface, application, website

Description automatically generated

**This is the**

**visualisation of the**

**training data.**

# How They Work

To make all the things work, and to upgrade the accuracy, I must augment the data. In data analysis, data augmentation refers to approaches that are used to expand the amount of data by adding slightly changed copies of previously existing data or freshly produced synthetic data from previously existing data. I augmented my data by adding some noise and blurring the first fifty images. After blurring I chose to create a new data set which starts with my augmented images and continues with the original images. To blur images, I used the OpenCV function medianblur(). The Median blur operation works in the same way as the other averaging procedures. The median of all the pixels in the kernel region replaces the image's core element. The edges are processed while the noise is removed in this procedure. medianBlur(src,ksize) where src is the image source and ksize represents the kernel size.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

The second approach was to reverse the image matrix. To do that accurate you have to change the x and y coordinates of the labels as well. For that, I use OpenCV and NumPy libraries.

**cv.rotate(src,rotateCode = x)** where src stand for image source and rotateCode(0= 90 degrees and so on,1,2) used this method to change the image with 180 degrees.

**np.rot90(arr),** arr stands for the array which we want to rotate with 90 degrees.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated **VS**

# 

# 

In addition, I decided to make some adjustments to the colour channel so that my model can train more efficiently. I utilised the NumPy library's **clip function ().** This strategy assists me in reducing the intensity of the colours. Because this method works well when combined with other data for training, I built a new vector of clipped photos and merged it with the original vector.

**Graphical user interface, application

Description automatically generated Graphical user interface

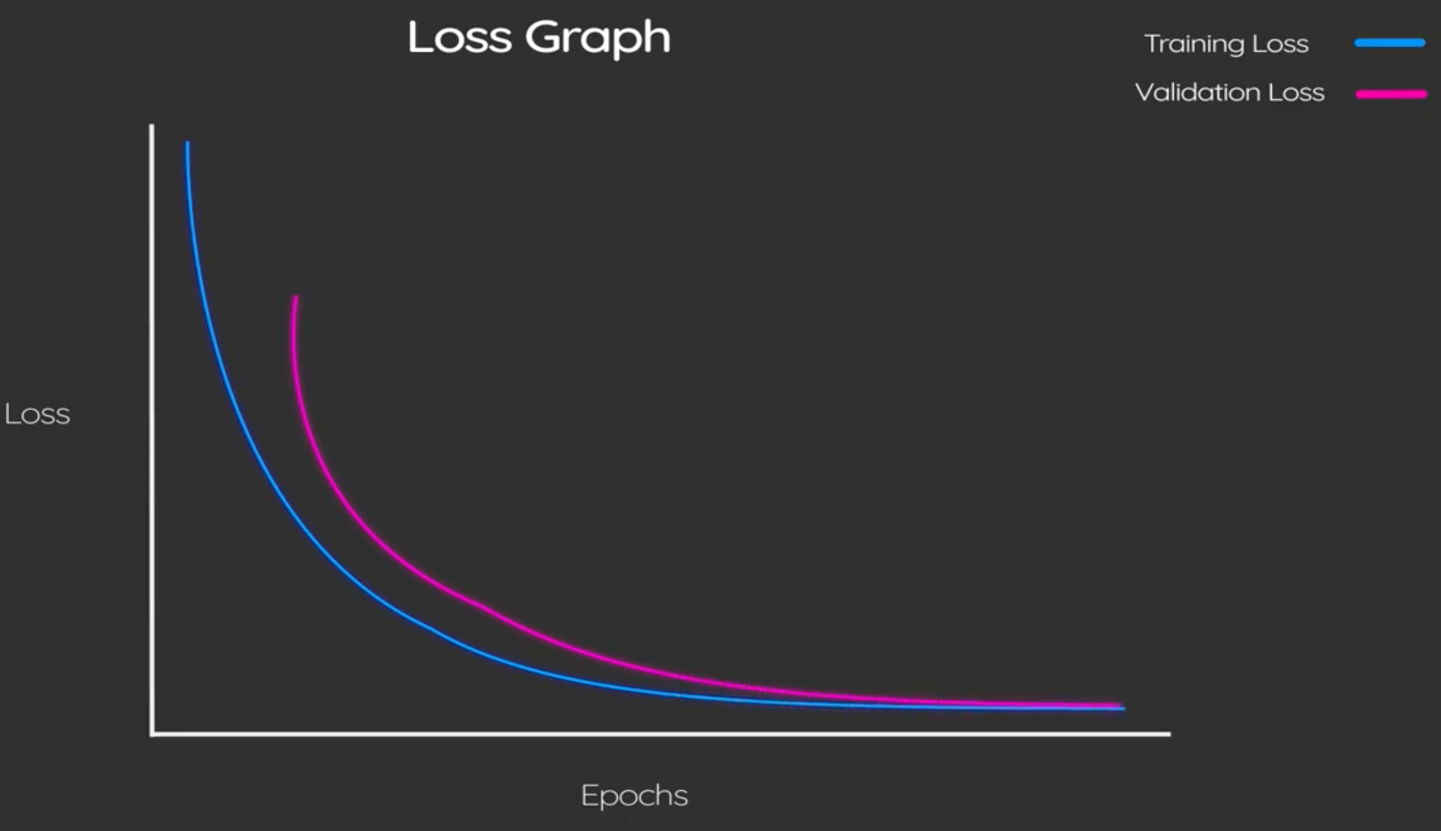
Description automatically generated with medium confidence**

I now must construct my train and evaluate sets once I enhance the data. For this report, I trained my model with various sample sizes to examine how it correlates with accuracy. I've seen that when more and more diverse data is included, I obtain a higher accuracy number, but at a certain point, it starts to overfit. Also, I must split the picture data to get it in the correct range and convert it to floating points so that my CNN has the best accuracy and dynamic range at the price of significantly more memory and processing time.

# Model

I will be coding the model with python using the Keras library. The architecture of my model will contain five major layers, one fully connected layer and a target output layer. Each of the major layers contains a convolutional layer and pooling layer of size 2 by 2. Now the first convolutional layer contains 16 filters, the second one has 32, the third has 64, the fourth has 128 and the fifth one has 256 filters. All these filters are of size three by three, and I will use relu activation function in these layers. Well after that the flattened values from the fifth major layer will be fed to the fully connected layer with 512 nodes then the values will be finally sent to the output layer with 84 nodes (42X2).

Choosing a certain layer setting is just a trial-and-error procedure that involves testing the dataset with several settings and selecting the one that results in the least loss and matches the data best.



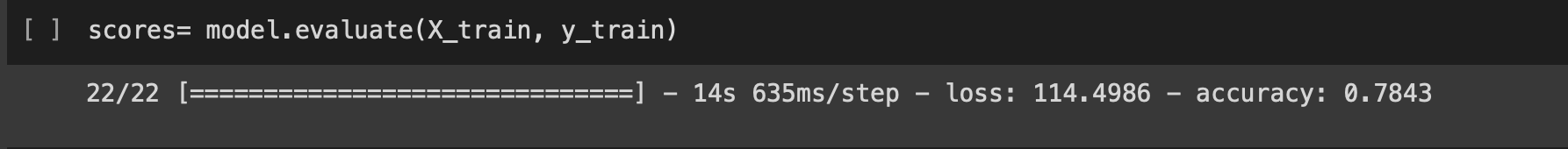
So, if we plot the training loss graph, we can see there is a steeper curve at first, and then both training validation and loss flatten off and almost converge, confirming that our model did not over-fit the training data but recognised the face characteristics successfully.

I trained my model 60 epochs with a batch size of 64. Also, I used Adam optimizer, mean squared error for loss and metrics accuracy.

Text

Description automatically generated

# Performance

My algorithm performs well, I trained it with original data, with augmented data and even I built another more complex CNN which was overfitting. In the end the accuracy of my predictions were situated between 78 % and 86% . The train set used for my report was of 700 samples and the validation set was 300 samples. My model performed with an accuracy of 78% percent but the predicted points on the example data were more on their actual place

that’s why I stayed with this performance to do the supplementary task for colour changing.

Graphical user interface

Description automatically generated with low confidence

A picture containing text, bottle, sign

Description automatically generated

# Additional Task

For the additional task where we had to change the colour channel, I used example sample to show how my method works. I changed the values of RGB to (250,0,0) which means that starting from the first pixel of the predicted landmark I change the colour in red till the last coordinates of the landmark. It is not accurate, but I can use histograms which calculates the intensities of the colours in that region and change the pixels with the highest intensity. Also for this task I could use segmentation and change the gradient for segmented area.

Graphical user interface

Description automatically generatedA picture containing text, indoor

Description automatically generated

# 

# Conclusion

In conclusion I used a 5-layer Convolutional Neural Network which trains on a sample data of images and key points, and it can predict with 78% accuracy facial landmarks on a random photo. For this task I realised that a lot of input data helps accuracy to grow but if it is too much data the model will start to overfit.

The results of this work demonstrate the influence of a set of optimisation and training procedures in the context of a face alignment system.

# References

* datahacker.rs (2020) #010 How to align faces with OpenCV in Python - Master Data Science, Master Data Science. datahacker.rs. Available at: https://datahacker.rs/010-how-to-align-faces-with-opencv-in-python/ (Accessed: May 12, 2022).
* Casado, C.Á. and López, M.B. (2021) Real-time face alignment: evaluation methods, training strategies and implementation optimization - Journal of Real-Time Image Processing, SpringerLink. link.springer.com. Available at: https://link.springer.com/article/10.1007/s11554-021-01107-w (Accessed: May 12, 2022).