Supervisor Dr Avgoustinos Filippoupolitis

Word

Count 12,

056

**A**

**DISSERTATION SUBMITT**

**ED IN PARTIAL FULFIL**

**MENT OF THE**

**U**

**NIVERSITY OF**

**G**

**REENWICH**

**’**

**S**

**BS**

**C**

**(**

**H**

**ONS**

**)**

**C**

**OMPUTER**

**S**

**CIENCE**

University Of Greenwich

An Investigation into how Neural

Networks affect facial

recognition with an

implementation in C#

Final Project Report

18

-

04

-

2

016

# ABSTRACT

The intention of this project it to investigate and successfully develop a neural network in C# for facial recognition. The network also has to be designed in such a manner that it will be compatible with other programs. This project report specifics all of the details within the design and implementation as well as discuses all of the issues which occurred during the development. It concludes on the overall effectiveness of the developed network.

## Acknowledgements

First of all I would like to thank the following people for their support and assistance during the course of this project:

* My supervisor, Dr Avgoustinos Filippoupolitis as well asDr Richard Anthony for supporting me through the project development and supplying valuable feedback relating to the development of the software as well as the overall structure of the report.
* My friends and family that have supported me through this assignment.

# CONTENTS

Abstract .................................................................................................................................................... i

Contents ................................................................................................................................................... ii

1. Introduction ..................................................................................................................................... 1
   1. Background Information ......................................................................................................... 1
   2. Scope of the project ................................................................................................................. 1
   3. Project aims/objectives ............................................................................................................ 1
   4. Methodology............................................................................................................................ 2
2. Literature Review ............................................................................................................................ 3
   1. Introduction ............................................................................................................................. 3
   2. Facial recognition: ................................................................................................................... 3
   3. Eigen face ................................................................................................................................ 4
   4. Neural Network ....................................................................................................................... 5
   5. Conclusion ............................................................................................................................... 6
3. Review of Similar products ............................................................................................................. 7
   1. Introduction ............................................................................................................................. 7
   2. Product 1 .................................................................................................................................. 7
   3. Product 2 .................................................................................................................................. 7
   4. Review Table ........................................................................................................................... 8
4. Design of a neural network for facial recognition ......................................................................... 10
   1. Introduction ........................................................................................................................... 10
   2. Declaration of requirements .................................................................................................. 12
   3. Functional Requirements ....................................................................................................... 12
   4. Non-Functional Requirements ............................................................................................... 13
   5. Basic concepts of a neural network ....................................................................................... 13
   6. Training process .................................................................................................................... 13
   7. Validation .............................................................................................................................. 14
   8. The design of the Neural Network for facial recognition ...................................................... 14
   9. Calculations of the neural network: ....................................................................................... 16
   10. Training the network ............................................................................................................. 17
   11. The data set of images ........................................................................................................... 18
   12. Data normalization ................................................................................................................ 19 5 Development of a neural network for facial recognition ............................................................... 20
   13. Introduction ........................................................................................................................... 20
   14. Feedforward ........................................................................................................................... 20
   15. Prototype 1............................................................................................................................. 20
   16. Prototype 2............................................................................................................................. 21
   17. Prototype 3/final network ...................................................................................................... 22
5. Programming ................................................................................................................................. 22
   1. Activation function ................................................................................................................ 23
   2. Feed forward calculations ...................................................................................................... 23
   3. The training process .............................................................................................................. 24
6. Legal, Social, Ethical and Professional Issues............................................................................... 26
7. Testing ........................................................................................................................................... 27
   1. Introduction ........................................................................................................................... 27
   2. Prototype 1............................................................................................................................. 28
   3. Prototype 2............................................................................................................................. 28
   4. Prototype 3............................................................................................................................. 29
   5. Summary................................................................................................................................ 30
8. Evaluation ...................................................................................................................................... 32
   1. Functional Requirements ....................................................................................................... 32
   2. Non-Functional Requirements ............................................................................................... 33
   3. Neglected consideration ........................................................................................................ 33
   4. Further development .............................................................................................................. 34
9. Conclusion ................................................................................................................................. 35
   1. Evaluation of the process ....................................................................................................... 35
   2. Evaluation of Software .......................................................................................................... 35
   3. Evaluation of Personal Performance ..................................................................................... 35
10. Appendix A ............................................................................................................................... 37

Part of the main class ............................................................................................................................. 37

1. Appendix B ................................................................................................................................ 43
2. Appendix C ................................................................................................................................ 44
   1. The data sets used for training prototype 2 and 3. ................................................................. 44
3. Appendix D ............................................................................................................................... 46
   1. Testing for prototype 1 .......................................................................................................... 46
   2. Testing for prototype 2 .......................................................................................................... 46
   3. Test nr.1 for prototype 3 ........................................................................................................ 48
   4. Test nr 2 ................................................................................................................................. 50
   5. Test nr.3 ................................................................................................................................. 57 15 Appendix E ................................................................................................................................ 59
   6. Images used for training ........................................................................................................ 59
   7. Images for testing .................................................................................................................. 59
   8. Images used within the final testing ...................................................................................... 65

16 Bibliography .............................................................................................................................. 69

# 1 INTRODUCTION

## 1.1 Background Information

Facial Recognition software although not that new is still in its development stages. It generally consists of comparing an image to faces stored in a database and the general aim is to match them. Those are very often used in automatic access control systems and human- machine interfaces. Such software is also becoming largely popular within registration software. Currently most schools and universities still use a sheet of paper or a card scanner in order to mark the attendance of a student. Both of the methods are faulty and the students don’t necessarily have to be there, in order to mark their attendance. This is an advantage of using a facial recognition software as it requires the student to be there in order to mark the attendance, as the person cannot ask a friend to sign them in. A fairly new way of creating facial recognition is with the use of neural networks. Although this project does not develop a fully functioning attendance system, the software discussed within the report is created with an intent to develop further into an attendance system.

## 1.2 Scope of the project

The purpose of this project is to develop and a neural network in C# which will be used for facial recognition, however this was not the aim within the initial report. The objective of that was to develop a fully functioning attendance system which uses a neural network for facial recognition for the university. As soon as further research had begun it became apparent that there were constraints when it came to the implementation of this system. One of them being that in order to create a whole facial recognition system one would need to firstly create a facial extraction program which can be used in order to extract the dimensions from features of the face which are used for input. Before they are used for input they have to be normalised therefore this would require another program of a similar size and difficulty level as the neural network and the extraction program therefore those were not implemented due to the lack of time. A revised set of requirements was created in order to serve as a guideline for this project. The project will have main two parts with one of them being training and the other being testing. One of the ways in which this project is evaluated, is based on the results of testing, while the other part of the evaluation is done by checking how well the program meets the requirements set.

## 1.3 Project aims/objectives

The aim of this project is to investigate the effectiveness of a neural network for facial recognition through a creation of such a network in C♯. Below is a list of the full aims and objectives.

Create a literature review

To research and critically evaluate similar systems

Gather training and testing datasets of images

Produce a design of a neural network which can be used for facial recognition in C# Implement a neural network for facial recognition, train it and test it.

Put the system through extensive testing and evaluate it by reflecting on the results.

Evaluate personal performance.

## 1.4 Methodology

Within the research for this venture many approaches were considered, like the waterfall approach and the evolutionary prototyping. However Round Robin has been chosen as the main development approach. The reason for this is not only the nature of this project but also the limited experience in neural networks at the start of the project. As this was a neural network multiple approaches had to be considered and then based on the trail end error the best method had to be chosen. The methods which had been considered were research truly and described in the literature review. The best way to describe this approach would be using sports. In sports the round robin means everyone plays against everyone and that way the winner is chosen. If there are many tasks which need to be completed at a similar time, just like within this project the Round Robin algorithm is perfect.

# 2 LITERATURE REVIEW

## 2.1 Introduction

The objective of this literature review is to view the literature published by others relating to facial recognition. Through reviewing how others have attempted to tackle similar problems an insight is given on different errors and problems they may have faced, therefore this might prevent the same mistakes being made within this project as well as this has given an insight on what methods are best to use when it comes to facial recognition as well as the approaches towards neural networks.

**2.2 Facial recognition:**

The first sort of any formal method used for this problem was firstly proposed in in a short paper written by Francis Galton (Francis Galton 1888). The paper itself did not provide any direct information on the best methods how to create such a system other than the use of using the multi modal classification which could be described as a semi supervised method for classification. At the initial stage neural networks were not the set technique for tackling the problem in the project therefore most of techniques have been considered. The best way to describe the approach within this paper is that it results in independent measures vectors which are then compared with all of the other ones within the database.  Facial recognition is an important research problem which spans over numerous fields given the nature of the problem. Face recognition is a biometric issue which is generally tackled by the use of automated methods which are then used with the persons physiological characteristics fingerprint, iris pattern, or face. Other biometric issues include biological patterns such as hand-writing, voice, or key-stroke pattern although since this software has been designed with an intention of being used for an attendance system characteristics therefore facial recognition seemed to be a more appropriate approach to tackle the problem. Within the paper written by Wenyi Zao (Wenyi Zao 1999) a three step procedure has been described which helped with creating the main approach of towards this project. The procedure is as follows:

1. A some sort of a sensor has to take an observation. This will create a biometric signature which will then be used as input. The type of input created will vary on the type of sensor used.
2. The next step will be the normalization of data, generally through some sort of a computer algorithm. The reason for that is so all of the data will be in the same format.
3. The third and final step will be the comparison of the normalized signatures with a sub- set on the systems database. This naturally provides something that’s described in this paper as “similarity score” which shows if the two signature are a match, and those are then applied to the user’s application of choosing.

Another interesting take on the advantages of facial recognition within the paper mentioned above is that facial recognition is considered a “non intrusive” system which is used to identify a person. Especially when compared to systems which use eyes for identification as people in general are very reluctant to use these sort of systems. Similarly within the survey by R. Chellappa, C.L. Wilson and C. Sirohey(R. Chellappa, C.L. Wilson and C. Sirohey, 1995) a two-step approach has been described. This begins with the face detection and normalization then the next step is face identification. The authors also described that the algorithms which are used to tackle the face recognition problem are split in to two types. Those are fully automatic and semi-automatic. The word automatic is used to describe algorithms which consists of the two steps while the semi-automatic only consists of the later half. In simpler terms this means that the fully automatic one creates its own input while the later one receives the input already prepared. Another interesting perspective in approaching facial recognition was provided within the paper written by Roberto Brunelli and Tomaso Poggio (R.Brunelli, T.Poggio 1993). The main way in which they are approaching this problem is through the use of unconstrained classification as the input of the system did not consist of a face. This was achieved by the use of templates of the eyes, nose and mouth.

## 2.3 Eigen face

While researching different approaches for facial recognition, it is only natural that one would stumble upon Eigen face. This is also known as the Karhunen- Loève expansion. This is found to be one of the more popular method when it comes to facial recognition The main argument within the paper by L. Sirovich and M. Kirby (L. Sirovich and M. Kirby, 1987) was that any image of a face can be reconstructed by the use of small collection of weights for each of the face as well as an Eigen picture. Furthermore the Eigen face algorithm which was motivated by Kirby and Sirovich has been thoroughly investigated within the paper written by Matthew Turk and Alex Pentaland (M.Turk, A.Pentaland). The database used by the authors had 2,500 images of 16 different individuals. Within the literature 96 percent, 85 percent and 64 percent of correct classification has been reported. Although it had been stated that those results have also been greatly influenced by the background as a large area of each image is the background and the lighting conditions. Within the same paper it was also mentioned that in order for the Eigen face approach to be used, an illumination normalization would generally have to be used. Within a journal L Zhao and Y.H. Yang (L. Zhao and Y.H. Yang, 1999) a method proposed makes use of three different images which have been taken under different lighting conditions in order to account for the arbitrary illumination effects. This particular piece of work has been built upon and taken further in the work 5 by taking the account of the Eigen features that correspond to facial components like eyes and nose, the authors used modular Eigen space which is composed of the Eigen features mentioned above. The literature concluded by stating that that a system based on this method would be less responsive in sensitivity of the appearance changes of the person then the standard version of the Eigen face method.

## 2.4 Neural Network

Just like with the Eigen face approach neural network are another popular way of tackling this problem and in makes cases both of these method are used combination with one another. A book written by Rogers & Kabrisky (Rogers & Kabrisky, 1991) discuss different type of architecture that can be used by the neural network such as single perceptron, the multilayer perception and the kohonen network. This book thoroughly discuses all of the mentioned networks and is definitely a great starting point for this topic. Although the book did not really offer any new perspective on the topic it did clearly highlights the point in which it is shown that a neural network becomes more “intelligent” through trial and error which can be described as training. Another paper which focuses more on comparing different types of neural network is a paper by Van Der Smagt (V.Smagt, 1990). This is a conference paper which discusses the advantages and disadvantages for every type of architecture which can be used for object character recognition (OCR). This is a similar problem to facial recognition and the discussion concludes by stating that a feed forward will be the most beneficial network to use for such a problem. This had a great influence on choosing the feed forward architecture to be implemented within the network used for this project. An interesting book written by T.J. Stonham (T.J. Stonham, 1984) provide insight on one of the first such networks which was used for facial recognition. The network discussed is WISARD which was a single layer adaptive network. It was also stated that one of the main attractive feature of using a neural network for such a problem was the none linearity in the network, as well as stating that the way such network is constructed is crucial in order for successful recognition to be achieved. The source also goes deeper into describing the best architecture for certain problems therefore stating that in the author’s opinion the best neural network for face detection the multilayer perceptron would prove to be most successful. Multi-layer perceptron being described as a feed forward network which maps out a particular set of input onto a set of ideal outputs, convolutional neural network can also be used for this sort of problem. Although the paper by Weng, J.S. Huang, and N. Ahuja (Weng, J.S. Huang, and N. Ahuja, 1993) suggest the use of a multi resolution pyramid for the face verification problem which although not used was greatly taken into the account when designing this project.

## 2.5 Conclusion

During this literature review the Eigen face method and neural networks were investigated as well as multiple other approach towards facial recognition. Within the Facial recognition part of this literature, concerns about the design and implementation were raised. Within the Source (R. Chellappa, C.L. Wilson and C. Sirohey, 1995) a two-step approach has been mentioned, therefore it had to be decided if the approach for this project will be fully or semi-automatic. Potentially this project was supposed to be a fully functional attendance system for the University. After the research was conducted it was decided that the fully automatic approach towards this problem is not an option given the timescale for this and the limited experience within this field. Although the Eigen face approach seemed attractive at first after further research into the Neural Networks it was discarded as that seemed to be the more “intelligent” option for tackling this problem. The sources especially helpful with reaching this conclusion are the paper by V.Smagt as well as the book T.J. Stonham as both authors clearly argued the advantages of the network to be used in solving this problem as well as gave ideas of what sort of architecture should be used.

# 3 REVIEW OF SIMILAR PRODUCTS

## 3.1 Introduction

The whole purpose of this section is to asses products which are similar in some ways to the softwere designed in this project. The review allows for an insight into the software’s capabilities and features. Although the programs discussed within this section did not necessarily have a direct influence on the design of this software, it did provide at somewhat of an insight into how the software might be taken further. Additionally a review table has been created in order to compare the existing products further.

## 3.2 Product 1

Google Cloud vision

This is an analysis tool which allows through encapsulation of powerful machine learning model to analyse image content and classify it accordingly. Although this software is not capable of facial recognition it is more than efficient at facial detection. The best way to view the appearance of this software is easy to use. Since at the moment only the Beta version is available, the API it accept base64encoded images only although it has been stated that within future uploads the image wont be required to uploaded to the API as the whole system will be integrated with the Google Cloud Storage therefore the system will be substantially faster than it is at the moment.

## 3.3 Product 2

BioID

This program has been created by a company called BioID Gmbh. Their program simply allows for biometric authentication integration for mobile platforms. This system works by allowing the user to authorise themselves through taking an image of themselves or a selfie (for a lack of a better word). The whole company has strong feelings about the superiority of biometric authentication rather than the traditional authentication through passwords. It has been mentioned by the CEO of the company Ho

Chang (Ho Chang, 2015) that even though a “strong” password is secure it can be somewhat troublesome to enter on a mobile device while the only user input which is required by this system is biometric one. The company has lately released a free login app for the IOS which is simple to use as all it requires is a camera on a phone and an IOS. There is not much to the API as the login system uses the second internal camera.

## 3.4 Review Table

The review table has been created in order to further compare the existing products. A review criterion had been created in order to complete the table and to fully analyse the products. As the products reviewed within this section are somewhat similar to the one designed within this project, most of the criterion within the table are designed to check if the product makes a satisfactory facial recognition system. Those requirements can be viewed below.

### 1. System should successfully detect a face

Before any facial recognition can be achieved, firstly a face has to be detected. The program has to successfully be able to separate a face within an image from the rest of the background in order to create some sort of an input.

### 2. System should use a type of machine learning for facial recognition

This makes any facial recognition system work more efficient and “smart”. In a perfect world a system would be able to use just one image of a person to learn and be able to recognise him, even if there are slight differences in appearance over time.

### 3. Usability

Is the software easy to use? A large majority of biometric authentication systems users could be describes as novices within IT, as they might my computer literate but their do not poses any advanced technical skills. This is why it is extremely important to make the overall software (especially the API) usable to users of mixed abilities.

### 4. “Runnability” and Responsiveness

Every program which is deployed should have a minimal amount of bugs and errors. Otherwise the software does not run smoothly and the companies which have deployed faulty program are most likely to lose users.

### 5. Privacy

Application which deal with facial recognition when deployed, have to deal with privacy issues as their databases consist of private information and images. Therefore companies have to make sure everything is done according to all of the privacy laws which apply to them.

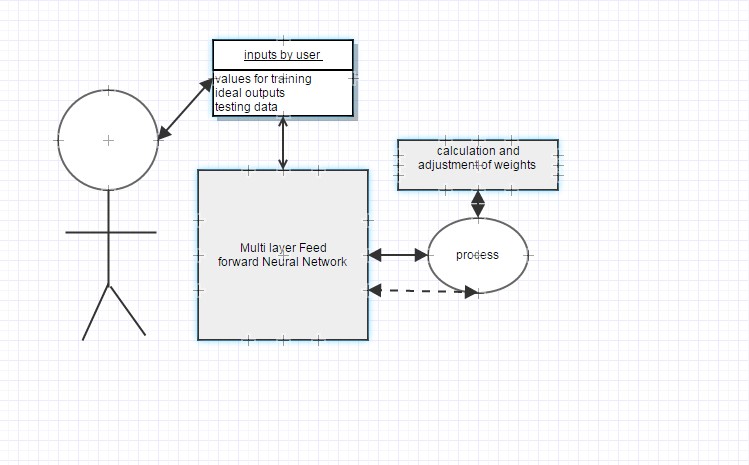
|  |  |  |
| --- | --- | --- |
|  | Google Vision | BioID |
| System should  successfully detect a face | Yes | Yes |
| System should use a type of machine learning for  facial recognition | No | Yes |
| Usability | Yes | Yes |
| “Runnability” and  Responsiveness | No | Yes |
| Privacy | Yes | Yes |

## 4 DESIGN OF A NEURAL NETWORK FOR FACIAL RECOGNITION

### 4.1 Introduction

This section provides a detailed design of the neural network which is being created within this project. It mentions some of the key aspects which had to be looked at in more depth before the implementation as well as provide the overall description of the calculations which will take place within the implementation stage of this project

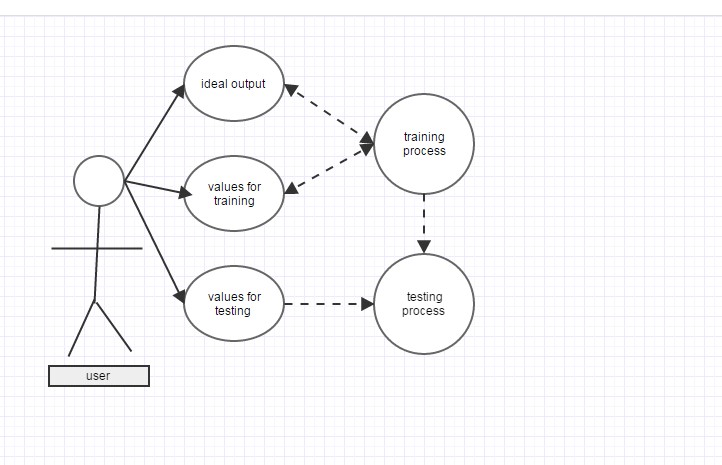
#### 4.1.1 Architecture Overview of the Neural Network (Rich Picture)



**Figure 1- the overview of the architecture designed within this system**

This is a rich representation of used to represent complex systems. When creating a rich picture one does not have to follow any sort of strict guidelines as it is ones representation of the system. In other words the person designing the rich picture can relate his own perceptions of the system within this diagram.

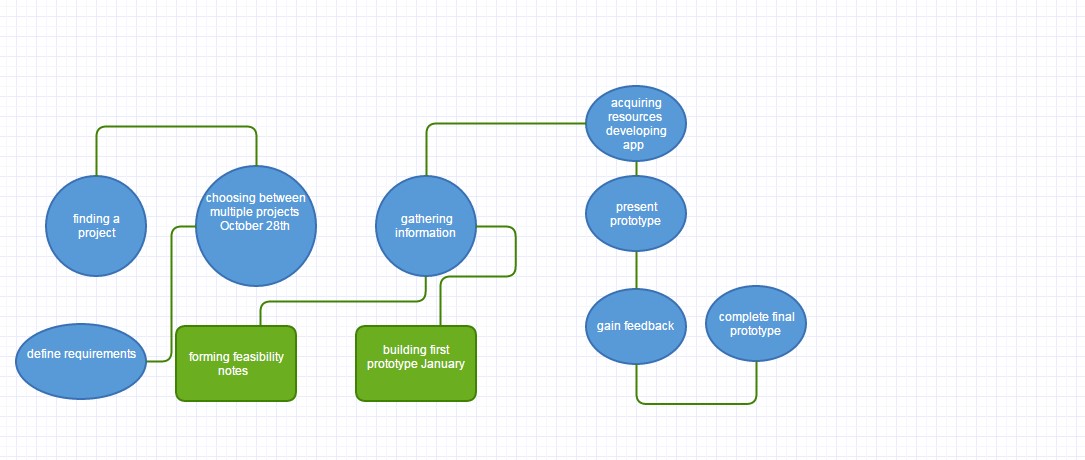
#### 4.1.2 UML Case Diagram



**Figure 2- this is a UML diagram for a Neural Network in its simplest form**

This allows for a graphical representation of the problem at hand. This can serve a basic blueprint for the programed worked on. Unlike the rich picture a UML is required to follow certain guideline.

#### 4.1.3 Critical path analysis



#### Figure 3- Critical Path Analysis

Similar to Gantt charts Critical Path Analysis serves as a basis for scheduling a set of projects happenings. The process a strict syntax as well as it forces one to follow certain guidelines. The Path above simply displayed the activities which have occurred within the creation of this project.

### 4.2 Declaration of requirements

The requirements set for this neural network intended to represent the functionality that the network targets to attain. They were chosen during the extensive research which has been conducted during the initial stages of this project. Through the research a set standard has been found which if reached within this project would suggest that the software has been successful.

### 4.3 Functional Requirements

1. The Neural Network will be successfully trained for the faces used within the training section.

The Neural Network should smoothly perform the demanded number of iterations.

1. The Neural Network will successfully recognise the person which it was trained even if a different photo of that particular individual is used.
2. The data within the network will remain private and will not be used for any other purpose other than as input for the network.
3. The software should not crash based on the system that’s using it.
4. The system should be able to handle possible errors and handle it accordingly.

### 4.4 Non-Functional Requirements

1. Software should respond well to framework updates.
2. Software should be capable of running on multiple platforms.
3. Software should be able to pull results as possible.
4. Software should be easily recoverable and reusable

### 4.5 Basic concepts of a neural network

Artificial neural networks could be described as an attempt to create a program with a ‘human–like though’ and are based on the biological neural networks, more specifically the human brain. Instead of modelling the human brain as a whole, the neurons are mostly taken into account as a basic building block of the brain. Therefore neural networks attempt to simulate the behaviours of those cells. The general concept of it is that when a neuron accepts a signal it may or may not fire. If it does fire the signal will be transmitted over to the neuron axon terminal after which it will be transferred to other neurons or nerves.

The ANN’s are structured in a similar manner. There is generally a minimum of three main layers to most ANN’s except the Hopfield neural network (which only has two). The first is the input layer, the second is the hidden layer and the third is the output layer. Each of the layers have connections with weights attached to them. So connections are called synapses and as well as the weights are what determines the Neural Networks output, therefore the weights can be described as the memory of a neural network.

Now if the weights are the memory, the training of the neural network would be the process in which the weights values are assigned.

### 4.6 Training process

Generally most training processes would begin by assigning random weights, after the values are assigned the networks validity has to be checked. After that the weights will be adjusted based on the results the network has produced and the error, after this the process will be repeated until the error is within the acceptable limit or the network has performed an acceptable amount of iterations and hasn’t reached the acceptable result therefore it has to most likely be re structured.

There are main two types of training a neural network and those are:

Supervised- this consists of providing the neural network with a set of sample data with a set of ideal outputs. This is probably the most common way of training a neural network, when the neural network starts training the neural network is taken through a number of iterations until the actual output matches the ideal output provided and has a reasonable small error. When it came to the facial recognition this was the method chosen considering the problem where the network has to recognise the face which was used as the input.

Unsupervised- this type of training a network does not provide expected outputs. This type of training is generally used when the network is used in order to classify the input. This sort of training also requires many iterations and during the progressions the classification groups are found. Therefore, if the aim of the network was to classify the faces of a human and animals this would be the training type which would be probably used.

There are also many training methods which combine the aspects of both supervised and unsupervised training with one such method being reinforcement training. During this method the network is provided with some sample data but it is not provided with ideal output data, just like with the unsupervised way of training, however the neural network is told whenever each output was right or wrong depending on the input therefore this training method would be a combination of both previous training methods.

### 4.7 Validation

Validation of a neural network mostly consist of further testing using more data. When it come to the neural network for facial recognition, the more images are used for testing the more reliable the network becomes.

### 4.8 The design of the Neural Network for facial recognition

Multilayer feedforward backpropagation neural network was the architecture which was applied in in order to create the neural network for facial recognition. This network being a feedforward network meaning each layer of the network will only be connected to the next one ( input layer to the hidden layer and so on). When it comes to the input neurons which could be done in a verity of ways in this project but there are the main two. The first using a facial extraction program and then normalising the data. It will start by the extraction of eight features of the chosen face which will become the input, therefore there will be eight inputs to the neural network. When it comes to the output this will equal the amount of faces used in the dataset and each output neuron will represent the face from the dataset therefore if there are 15 people in the dataset there will be 15 output neurons. As this system is developed with a purpose of being used as the university’s attendance system another way of creating such a network would be with the use of the student ID card. There would be only two input neurons and only one output neurons.

The two inputs would check if they are a match by being fed through the network and outputting through the single output neuron if the faces matched or not. Within this particular project, first design has been chosen and implemented. Generally there can be any number of hidden layers within such network as long as there is at least one hidden layer (without the hidden layer the networks architecture would no longer be a multilayer feedforward neural network). Within this network only one hidden layer exists as this is mostly sufficient for majority of the problems as adding more and more hidden layers does not necessarily improve the performance as this might cause over fitting(this means that the network fits the dataset perfectly but when a new dataset is used for testing the performance is poor) .

There is no general rule when it comes to the amount of neurons within that layer (there are suggestions such as the mean of the input and output layers there) therefore the amount of hidden neurons which this network will begin with is five. Each of the layers are also going to have bias neurons but the bias neurons will never be connected to the input neurons. The value of the bias neurons will always be set to 1.

Now as this is a three layer neural network there will be two weight matrixes which occur between the connections of the neurons. Below there is as diagram of a feed forward neural network that’s going to be implemented.

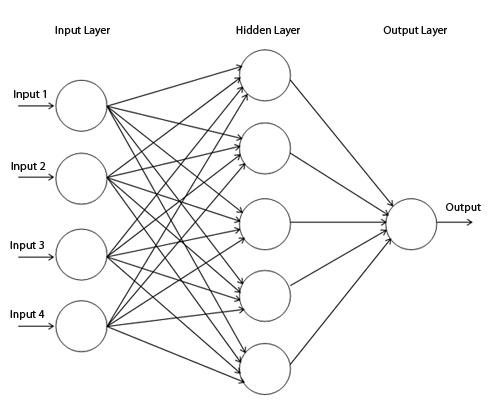


Figure 4- this diagram represents a multilayer feedforward neural network which is implemented within this project

**4.9 Calculations of the neural network:**

In order to calculate for the output value for each of the neurons the following formula has been used:

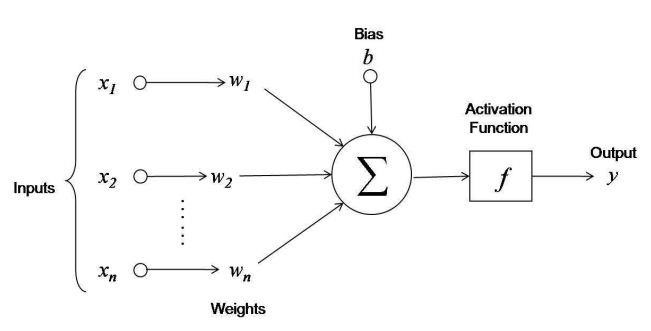


Figure 5- this diagram shows the calculation of the neuron within the network

Take the input value and times this by the weight values, this is done for every input and weight value connected to the specific neuron and then those values are added together with the bias value. Now this whole thing will be thread through an activation function (which in this case is the sigmoid activation function which will be described later in this report) and the output of the neuron is calculated. This is then repeated for each of the neuron until the final output neuron is reached.

### 4.10 Training the network

For this project the backpropagation algorithm was used, this is a supervised method of training a network. This training method specifies that the delta weight for the weight t is equal to the epsilon (this is the learning rate which specifies how quickly the weights will be updated) times the partial derivative of t with respect to weight t plus the alpha (this is the momentum which is there to escape the local minima) and then times the delta of t minus 1(meaning this is the previous change from the previous iteration). Now this equation has to be applied to the gradients. Now the two parameters which are the epsilon and alpha respectively and their values have to be picked before inputting anything into the neural network as this will have a great effect on how successful the network is at training and the values for those training parameters generally range between 0.1 to 1. The appropriate values will be put through this equation and the value which comes out is the change in weight and this will be added to the old weight therefore producing a new weight for the synapses, therefore completing an iteration of the training process. Now in order to successfully train the network, some sort of a percentage needs to be calculated in order to see how successful the network is becoming after each iteration. This will be the error rate and the purpose of training is to minimise it. In order to complete the training of the network this error rate has to be calculated after every adjustment of weights. There are many ways in which this error can be calculated but the one which is going to be used for this particular network is the Root Mean Square (RMS). The formula for this calculation shown below:

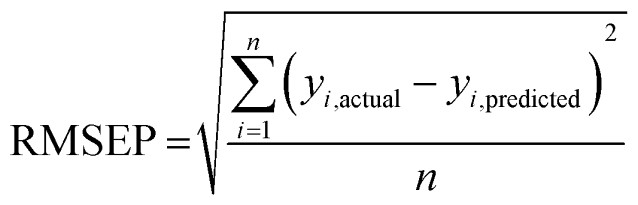


Figure 6- this shows the mathematical calculation for the Root Mean Square

So in order to calculate this the actual output has to be taken and subtract the ideal output from it and then square it, for the next actual and ideal output the same thing is going to be done. Then everything is going to be added together. The next step is to divide this number by the number of outputs so if there is an nth number of outputs the values will be divided by N. Now in order for this to be an RMS instead of just a Mean Square Error calculation (MSE) this whole equation has to be fed through a square root and the RMS is calculated.

When the error is calculated the weights have to be adjusted in order to further minimise it but before backpropagation can be used for this, the gradients have to be calculated. Now the deltas have to be calculated before the gradients can actually be found. The delta will be equal to the negative error rate, this is then timed by the activation function (in this case this is the sigmoid) where the sum of the inputs (those are the inputs from other neurons and values of weights added together) in particular neuron are being thread through it. Now to calculate the gradient for each of the synapsis the output of the hidden neuron is taken and timed by the delta of the output neuron and the gradient is found.

Now in order for this training to be successful the error has to be minimised as much as possible and a certain number of iterations when it comes to the training process has to be performed. Within this project the network will perform 10000 iteration before the training is complete or the error is reached.

### 4.11 The data set of images

In to create the input values for the netowork measurement of facial feature had to be taken by feature extraction program in this particular case it was the Face Mixer program from Abrosoft.

The data set which will be used at the beginning stages of the network consists of 15 different football players with 5 different images of them. This amount of images allows to extensively view the scope of how successful the network is. Although more images would be better this is a sufficient amount in order to state the efficiency of the networks architecture. For full datasets of the images used please refer to Appendix E.

In order to create an input from the images the features of the face have to be extracted from the player’s faces. For this particular there are 8 distances used and those can be found below.

Features used for extraction:

1.distance between middles of the eyes

2.distance between middle of the left eyes and middle point of mouth

3.distance between middle of the right eyes and middle point of mouth

4.distance between middle of the left eyes and middle point of nose

5.distance between middle of the right eyes and middle point of nose

6.distance between middle point of mouth and middle point of nose

7.distance of middle point of distance 1 and middle of nose

8.width of nose

By using those feature measurements this network in theory will still recognise a face even when someone will grow out their hair or a beard as long as the distances don’t change. Theoretically the problems will only start to occur once those distances between the features change. What this means is that if the network is trained using the face of a child, once the child grows significantly the network is most likely not going to be able to recognise it, as the distances between the features would have changed. Therefore, this network should work better with adults as they have stopped growing or when it comes to young adults as they will not grow by much and the network should still be able to recognise them.

### 4.12 Data normalization

Before those distance can be used as input the data has to be normalised. Normalization is process of correction data values in order to be in interval 0-1 using



Figure 7-this shows the formula used for Normalization of the values which were extracted from dataset

X - value that should be normalized

Xn - normalized value

Xmin - minimum value of X

Xmax - maximum value of X

The idea of preparing the data in such a way came from viewing the Neuroph project for facial recognition. Neuroph is a java neural network framework which can be used in a verity of ways and although the network itself was not used within this project it has given plenty of perspective on how the networks architecture can be structured.

## 5 DEVELOPMENT OF A NEURAL NETWORK FOR FACIAL RECOGNITION

### 5.1 Introduction

This section of the report mainly focuses on the implementation of the neural network in C# as a console application, as the problem tackled is quite complex it is a lot easier to firstly create a console application. The program is constructed in such a way that it can simply be converted into a fully running program with a GUI. During the development stage there were 3 prototypes created. In each of the prototype sections main purpose of each prototype is discussed as well as the main changes between them but when it comes to the actual programming of all of the programming approaches to calculations are discussed in the programming section of this report as the majority of those stay the same for each of the prototype.

### 5.2 Feedforward

The network itself is a feed forward with the exact specifics being described in the design stage of the report therefore within this the focus will mainly be on operations and their implementations within the code. Overall there were three prototypes of the network created although they only differ slightly, as the first prototypes was mainly created for training the network while the second also implements the testing of the network. There were many changes made in the both prototypes such as the change in the number of iterations used during the training process and the amount of neurons that were applied to the network. Those changes were significantly changing the output produced by the network and through mostly trial and error the best version was chosen. The way this network solves the face recognition problem

### 5.3 Prototype 1

This prototype mainly focused on the development of a simple neural network in C♯ which didn’t necessarily have to have anything thing to do with facial recognition. Neural networks are generally programmed in python or java with C♯ having very little resources available therefore creating this prototype in order to see if it was even possible within the timeline given was critical for this project.

The XOR problem seem like a good starting point and there is lots of information on implementation for a neural network even though it might not be necessarily in C♯. By looking at program written in python and java an implementation in C♯ was possible. When it comes to the amount of the input neurons only two are necessary in order to tackle this problem as well as one output neuron. Now when it comes to the amount of hidden neuron as mentioned in the design stage of the report it is always a bit of a guess work but during gathering the research for this project it has been found that three hidden neurons should more than suffice for such a problem. After the network was created all that was necessary was 2000 iterations in order to successfully train the network. The learning rate and the momentum for this particular prototype were 0.7 and 0.9 respectively. Overall this prototype has given the bases of the neural network for facial recognition as the all the calculations need by the network were already made within this prototype therefore it has minimized the amount of changes needed to be made in order to tackle the face recognition problem. The results of this network are within the testing section of this report.

### 5.4 Prototype 2

Upon the completion of successfully solving the XOR problem within this prototype the focus was shifted on making the network successfully able to recognize faces. One of the main differences was the data used and its preparation( all of those are discussed within the programming section ). The values within the arrays used for input and the ideal output also had to be changed and expanded. The values which were extracted from the images and then normalized were used for the input array and the array used for ideal output had to be expanded and formed in such a way that each of the 15 outputs are going to correspond to each of the image of the football player used. To view the data used in the new arrays please refer to Appendix C. Other than the input the amount of neurons had to be changed as this is a different type of the problem then the XOR. As mentioned within the design stage there will the amount of data within those array directly corresponds to the amount of neurons that will be in each of the layers. Therefore the input layer will have 8 input neurons (this corresponds to the amount of features used for input) and the amount of output neurons will be 15(the amount of football players used). Like discussed within the design there will be 5 hidden neurons within this prototype. As this problem generally requires more data to be used naturally more training iterations will be required in order for the network to learn. The amount of iterations have been set to 20000 as it seemed like a sufficient amount. Based on the results produced by this prototype adjustments were made in order produce a network which successfully recognizes the all of the faces that it has been trained for. The results of the testing can be viewed in Appendix D

### 5.5 Prototype 3/final network

Based on the results from which the previous prototype has produced. The network had to be adjusted accordingly. The number of iterations has been chosen to be 15000. This proved too much, the trained values were recognized perfectly but when new data was entered into the network the images were no longer recognized. Therefore, as the network has run into over fitting. This meant the amount of iterations had to be lowered in order for the network to recognize the images of the same people but different photos. After trying out different values for iterations the value has been set to 10000. The neural network was not quiet converging as the weights were getting updated too quickly. Therefore the learning rate had to be dropped even lower than before. A more convenient way of tackling this problem would be through the use of resilient propagation(another way of training a neural network) were those values do not have to be picked. Although the majority of things within the network were adjusted the network was still not producing the right values. This could’ve meant that the networks overall architecture had to be evaluated once again and possibly changed, but before this happened the amount of hidden started to be changed and when it was finally lowered to four. This seemed to have solved the problem. Arrays used for training, and initial testing stayed the same within this prototype as within number two, but additional data has been added in order to further test the network. In order to view the test data and results please refer to Appendix D

## 6 PROGRAMMING

### Main class

The class mostly contains the arrays which are used for training and validation. Those arrays are then put through the training methods and later used to train neural network. Now there are two arrays one used for training with the other one being ideal. Those arrays are simply called input and validation and are two dimensional double arrays. If by design the neural network only had one output neuron, a single dimensional array would certainly suffice but since this problem requires more than a single neuron to be used this will no longer be enough. Well the next step within this class was the creation of an object called FeedforwardNetwork

FeedforwardNetwork network = new FeedforwardNetwork();

Now the next step will be the creation of the layers which will be added to this object in such a way: network.AddLayer(new FeedforwardLayer(8));

Now the first layer added will be the input layer, the second is the hidden layer and the third is the output layer. If an additional layer was added in the middle then it would have become the second middle layer as always the last layer specified is the output layer.

Once the required layer are added to the Neural Network the training object is created. Now this object requires several objects to be passed through to the constructer. Firstly its specifies the object which was created in the step before (this is the network which is going to be trained), the next are is the training set(the input array) and the expected output and those will be used in order to train this network in order to solve the problem that is facial recognition. The learning rate and momentum are also specified ( those are described in the design section of the report).

while ((epoch < 10000) && (train.Error > 0.001));

Here the amount of iterations is specified as well as the error that would be acceptable for this network, therefore this program will perform 10000 iterations and then stop or it will stop before if it reaches the error. Once the network stops looping through the iterations it will output the results

### 6.1 Activation function

As specified within the design section this network uses a Sigmoid activation function. This function is specified within the Activation Function interface. The interface also specifies the derivative function as this is necessary in order to for the network to implement Backpropagation (this training method cannot be used on a network which uses an activation function without the use of derivative) which is going to be used to train the network.

Those methods are simply mathematical equations (the specifics of them are described within the design part of the report) and are shown below

public double ActivationFunction(double d) { return 1.0 / (1 + Math.Exp(-1.0 \* d)); } public double DerivativeFunction(double d) { return d \* (1.0 - d); }

### 6.2 Feed forward calculations

Matrix operations are used within the network in order to do calculations within the network.

Now the input array will be used in order to populate a matrix and another row will be added where all of the elements of it will be equal to one (those are the bias values). Within the network a Compute output function was created. One of the first things which this method is going to do, is going to check if the input array. The size of it must much the size of input neurons within the input layer.

if (input.Length != this.inputLayer.NeuronCount) {}

Then the objects are looped across, the input layer receives the values from the input layer while the hidden layer receives the output values of the input layer as input and the same goes for the output layer which takes the output values from the hidden layer and uses that for input.

The feed forward layer class contains the Compute outputs method which is used for the calculations for the network. The first thing which this method is going to do is to check if the pattern has been presented to the network. The pattern will also be copied to the fire variable, the next step would be to calculate the output for each of the neurons. Now in order to do this the column matrix must be obtained for each of the neurons in the next layer. Now the dot product between the weight column matrix and the input must be calculated as that is the value which will be presented to the next layer. The next two steps will be passing those values through the activation function and then storing those values as the fire variables of the next layer, and the output form the output layer is the output for the whole neural network.

### 6.3 The training process

As discussed during the design stage of this report the training method used for this neural network is backpropagation. As discussed previously after the output of the iteration of the neural network has been calculated the weights must be updated accordingly based on the results of the iteration. Therefore the backprogagation algorithm is used in order to update the weight and the threshold matrix therefore make the network “learn”. When is come to the “learning” of the network there are two main classes which are employed. The backpropagation class which holds and uses the Train interface(which carries two properties and one method). The class also uses the backpropagation layer class which is used to hold information which are then going to be used for each of the layers of the network. The method inside the backpropagation class is called iteration and it is responsible for performing each of the iteration of the training within the network. The property called network holds the latest and most efficient version of the neural network and the error property holds the current error of it. Now before the error can be stored it has to be calculated. The method called CalctError from the backpropagation object and firstly it will ensure that the ideal output array matches the number of output neurons. The algorithm then will propagate backwards but before this can happen objects and the errors from the previous iterations of the backpropagationlayer must be cleared. Within the backpropagation layer there are two versions of the CalctError method. The main version is responsible of calculating the error only for the output layer, this is done by calculation of error shares for each of the neurons and then looping over all of the output neurons.

Now the error has to be set for the neuron (the error in this case will simply be the difference between the ideal and the actual output). Then lest step will be making the numbers bound so they will not become too large or small. All of the other layers within the network will have their error calculated by the other version of the CalctError method which doesn’t use an ideal array. Now every matrix connections between the layer which has just been calculated and the next have to be looped through and like with the previous method the deltas of the matrixes have to be accumulated before the errors are cleared out.

AccumulateMatrixDelta(j, i, next.GetErrorDelta(i) \* this.layer.GetFire(j));

As the previous stage was all about the calculation of all of the necessary error calculation this stage is all about applying those changes to the network. This starts with all of the backpropagation layer objects are being looped over with the learn method being called for each one of them the changes are applied. The way the learn method work is firstly making sure there is a matrix to be used. The matrix contains all of the delta values calculated previously and being scaled by the learning rate(the importance of the learning rate has been described in the previous section). Then the momentum also has to be applied and both of them added together and put in the matrix delta variable. Then the calculated values are added to the current matrix and this causes the learning to occur, before the next iteration can occur all of the errors have to be cleared.

### 7 LEGAL, SOCIAL, ETHICAL AND PROFESSIONAL ISSUES

Legal issuses assoitated with the cration of this neural network will depend on the application of this network, as this project although initially intended to be a whole fully functioning attendance system for the university of Greenwich. As of now its only a artificial network network which potentially can become a part of the attendance system. Therefore this whole project should be thought of as a prototype so most of the legal issues within the field of technology which would apply to it if this was a part of an attendance system do not apply to it yet. Nonetheless they will be discussed within this section of the project. The biggest legal issue using any facial recognition system when deployed is using personal data, therefore the Data protection agreement will come into play as images of people registered in the system will be stored . This Act has been launched in March of 2000 and the purpose of it is to keep any personal data or information safe and not shared or processed without their consent or knowledge. In short there are eight main principles which a company would have to follow in order to abide by the Act.

In other words the issue which is talked about here is Confidentially. Generally speaking people are very reluctant when it comes down to sharing information, intellectual ideas property especially when they know that those will be used in some technology. The act above was created to ensure that, as was the Non Disclosure Agreement This is especially useful when dealing with companies coming in to work on a project as they would be forced to sign a non disclosure agreement so they will not give away any confidential information.

When creating any sort of project within the field of IT it is important to remember that there are standards which are being set out by professional bodies such as BCS/ACM/IEEE. Although it is not required to be a member of any of the organisations mentioned above in order to work in IT. Following it is always good practice to as it gives a certain level of professionalism within the work. Within the public interest section of the code of conduct it is stated that professionals have to have a regard for privacy, security and wellbeing of others. In terms of this project if this software was to be deployed it would mean that all of the necessary steps would have to be taken in order to ensure those things such as the data of the student is protected as well as prevention of any potential malicious software from being uploaded and therefore putting the security and privacy of students as the number one priority. Another part of it is a statement that a professional should conduct his activities without any type of discrimination. Now here, things like making the software accessible and usable for most users is very important. Here the use of this software is by students as well as the lecturers with different disabilities would have to be considered. This is also considered a social issue

Naturally when developing any kind of project especially when it comes to deployment ethical issues have to be considered. Those would include the treatment of data as if it was your own. In the case of this project the data would belong to students and it would be their photos, and some personal details about the person that the photo belongs to. When developing any sort of application, one must take into consideration if the system deals with vulnerable people or sensitive topics. Fortunately the only data which would be included in the system will be from the students. Another ethical issue which are taken into consideration is plagiarism. Research has to be done at the beginning of every project in order to gather information about projects and topics which are similar, this essential in order to give live to new ideas. The information used cannot be just simply used without the appropriate steps taken beforehand, otherwise this will be considered plagiarism. Honesty this is considered another ethical issue which in this case would require the data which was gathered from students not be used for any other purpose other then what has been stated beforehand and acquired their permission.

## 8 TESTING

### 8.1 Introduction

The tactic which was applied for the testing of this project mainly consisted of comparing actual with the expected output. Within this section all of the results from training and testing will be shown and discussed. All of the tests which were performed on this Neural Network were conducted in order to validate the networks functionality as well as to look for different aspects of the network which should be worked on in the future. The results from training will be separated by prototypes. There is aslo a final result section which were found using the final prototype of the network. The success is measured by how well those results are matching with the expected output. The second test which was performed on the final prototype consists of testing the network with images of people who were not used for training. The last test which was performed on the final prototype consisted of photos of people who were not used for training of this network. This test will further the effectiveness of this network and unlike in the previous testing, the success of this test is determine by the negative results. To further observe the in depth testing result please refer to Appendix D.

### 8.2 Prototype 1

Like mentioned within the implementation part of this report, this prototype was mainly directed into checking if a neural network(multilayer feedforward architecture) can be constructed in C# within a specific timescale. The XOR (Exclusive Or) problem is a logical operation which outputs true only if the inputs are different. The XOR problem seemed like an appropriate place to begin as it’s a common starting place for neural networks (this was found within the research) to start. The testing itself consisted of training and testing the network with the input values which were used for training. What can be seen from the results is that the network was successfully trained for the XOR problem. No further testing was done for this prototype as this particular network has served its purpose which was a successful creation of a neural network in C#. By looking at the results a clear match can be seen between the actual with the expected outputs therefore according to the testing criterion this prototype can be deemed a success.

### 8.3 Prototype 2

Within the testing designed for prototype 2 the focused has turned back to facial recognition. As described within the implementation stage of this report this network mostly aimed at making the changes necessary for this prototype to tackle the facial recognition problem. Testing although always an important part of creating a prototype, in this case it wasn’t necessary result oriented. As after making the changes from the XOR network it was quite clear the results from testing of this prototype will not be right but thanks to those, areas in which the network still needs reworking were found and applied in prototype 3. Similarly to the XOR testing conducted before, the testing for this prototype consisted of the input, ideal output and the actual output. Like mentioned within the designe section of the report if a network is trained successfully the actual output will be between a 0.9 and a 1. As most of the results above do not fit within the stated boundaries, it cannot be said the testing within the prototype was a success. Although the testing itself might not have been ideal, a pattern has been spotted within the network output. Most of the neurons went over the upper boundaries (in the case of this network this is the value of), generally when a neuron is too well trained for a particular value it will most likely going to cross over the upper boundary. Although most neurons have not reached the desired values, some have. Therefore it can be said the network although not validated, is on the right track.

### 8.4 Prototype 3

This prototype was the final one therefore it was pivotal for this testing to be a success as it would determine if the network designed is feasible. In the case of negative results from testing, the network would most likely have to be restructured. Similarly to the test for the prototype before, first test consisted of testing the network with the values from the photos that the network was trained to recognise. It is clear to see that the results returned by the network are all within the boundaries therefore it can be said that this network was a success. Although this testing was a success, the network itself is not yet validated as this test only proves if the networks training was successful. The validation of the network is proved in test number two of the final prototype. This is where the network is tested with the same players which were used for training, but different photos of them. This means that the input for the network will be different and therefore it will prove if this project is truly feasible. The last test which was conducted consisted of photos of people not used in training as input, this was done to further prove the networks overall functionality.

The first test produced the expected results and proved the network success in training. All of the values were between the values of 0.9 and 1.0 therefore they have met the testing criterion. In order to validate the network test number 2 was conducted. The test itself produced mostly positive results except for some iterations for some neurons. Each neuron has been tested using five different photos therefore as long as the majority of iterations returned a positive result the test for that particular photo was determined a success. That neurons which missed the expected boundary did not do it by much therefore they most likely could be fixed by changing the learning rate. To determine by how much the learning rate would need to be changed as well as the validity of the previous statement further testing would be required. Unfortunately the given timescale has prevented further testing form being completed. Within test number 3 most of the results miss their boundaries expect for two photos. After further investigation it has been found that the photos in question are quite similar ( to two photos within the main testing) in terms of the inputs which they produced therefore even the result was not desired it could’ve been

predicted.

### 8.5 Summary

Test performed on prototype number one although not extensive, produced the expected results and therefore created a base on which a neural network for facial recognition was been designed on. Within testing for prototype 2 many faults were found with the initial approach to solve the facial recognition problem. The testing did not output the desired results and overall it was determined a fail. Although not successful, like mentioned before this was expected as the purpose of this network was to create an initial approach for facial recognition and then build upon it in ways which were determined by the type faults found in the testing. For prototype 3 testing number one was an overall success as was testing number two. Although some faults were found a more then substantial amount of positive results was found in order to deem the network successful. Within the final testing of this network majority of the results have met the testing criterion. This however cannot mean that the tests aim has been achieved. This is because not enough photos were used within assessment to be deemed reliable as the timescale did not allow it. Overall majority of the testing have met its criterion, more importantly test number 2 from the final prototype was successful therefore the network was validated.

.

# 9 EVALUATION

The way in which the success of this system has been evaluated was previously discussed within the scope. The first criteria was for the network to produce successful testing results while the other was meeting the set requirements.

## 9.1 Functional Requirements

1. Neural Network will be successfully trained for the faces used within the training section. Requirements has been met as the network was successfully trained. This is confirmed within the testing section of this report.
2. The Neural Network should smoothly perform the demanded number of iterations. The network performs any demanded number of iterations smoothly. Different number of iterations were performed and documented within the implementation and the testing section.
3. The Neural Network will successfully recognise the person which it was trained even if a different

photo of that particular individual is used.

The requirement has been met with test number 2 of the final prototype. This can be viewed within the testing section of this report.

1. The data within the network will remain private and will not be used for any other purpose other

than as input for the network.

The intention behind designing this particular network was never to use the data for any other issue then what was previously stated therefore this requirement has also been met.

1. The software should not crash based on the system that’s using it. The software is written in C# therefore it performs smoothly on Windows and Linux but its abilities cannot be demonstrated on Mac OS. This means that the software only partially meets this requirement.
2. The system should be able to handle possible errors and handle it accordingly. Although not many exceptions were met during the creation of this prototype when it did occur they were handled appropriately.

## 9.2 Non-Functional Requirements

1. Software should respond well to framework updates. Within the time in which this project was created there were no updates to the framework in which the software was developed. However the network has been tested in the older versions of the framework and it performed just as smoothly as it did in the current one therefore this requirement has been met.
2. Software should be capable of running on multiple platforms.

The system performs smoothly on both Windows and Linux but it is not capable on running on Mac OS

1. Software should be able to pull results as fast as possible.

Unfortunately there were no speeds to compare the speed of calculating results against. However it can be said that the results are pulled as fast as the framework allows it too.

1. Software should be easily recoverable and reusable.

Within the creation of this project no events occurred where recovery of the source code was needed. When it comes to reusability the network was designed with the intention of using it in an attendance system for the university therefore the software was written in the most reusable way possible. To see this please refer to appendix F.

The project has been deemed successful as within the section above it has been show that the majority of requirements set were met. The testing part of this project also proved successful and the details can be viewed within the testing section of the report. The training process of the network went quite smoothly although it should be noted that the training algorithm could be improved, especially when a higher amount of people would be used for training the network.

## 9.3 Neglected consideration

This part of the report describes all of the issues which did not present themselves at the start of the project as well as the requirements which were not met. Like mentioned before the main issue which occurred within the creation of this project was not being able to meet the original requirements. This was not possible due to the lack of time not not possessing the right amount of knowledge. After the requirements were revised and the network began to develop the choosing the right training algorithm for the network proved especially difficult. Within the research for the project it has been identified that the genetic algorithm would most likely prove to be the most efficient. However during the same research that the genetic algorithm would’ve be a lot more difficult to implement as well as more time consuming and taking the amount of knowledge at the start of the project, the back propagation algorithm has been chosen in order to train the network. If the network were to be implemented into the attendance system this would most likely have to be changed as the network would have to be as efficient as possible. Taking this into further consideration more testing would have to be done. Within this project a sufficient amount of testing has been done but in order to check the network feasibility for the attendance system higher number of images and iterations would have to be used.

## 9.4 Further development

This project can certainly be taken further and the step to that would be through creating the attendance system for the university with the neural network created in this project. This would have to begin with the creation of a facial extraction software which would be connected to a camera. Ideally the camera used would be fitted on the ceiling of the lecture hall or a classroom and it would be able to scan a person’s face and then move on to the next one. This approach would save the students to have to come up to the lecturers desks in order to scan their faces by the camera in the computer. After the dimensions of the face are taken, they will be passed to the input normalization software, and after that is done neural network created for this project would receive them. Once the results are calculated another system would upload them online in order to mark the student’s attendance. The network would most like also be adjusted to fit the high volume of students.

## 10 CONCLUSION

### 10.1 Evaluation of the process

When writing the proposal for this project. Plans were made in order to monitor progress and ideally the progress made on weekly basis was supposed to be compared to those plans as a way to monitor overall progress. Now this was supposed to happen on a monthly basis but after the star t of the project this only occurred within the first two months. Although the progress was not monitored like it was meant to, this did not cause any major delays as the majority of time was spent on completing the project. Even though the work was completed on time there were points in which the project started falling behind. If the plan to monitor progress was carried on those minor delays could’ve been realised and dealt with earlier. Fortunately the time which was supposed be dedicated to monitoring the progress was not wasted as it was spent on development of the project and fixing the errors within. Since only a limited amount of knowledge was possessed about the topic at the start of this project, a lot of time was spent on research, this was then documented within the literature review. Choosing the right material was especially tricky as neural networks for facial recognition is a relatively new subject and there is simply not that much material about it especially when it comes to programing it in C#. Most of the materials on programming approaches for this problem would be in either Python or Java.

### 10.2 Evaluation of Software

This project has been completed on time and was deemed a success. Although the requirements had to be revised after the initial contextual report as during the research it was deemed not feasible within the given time scale. After those requirements were changed the network proved to be able to be comfortably trained as well as tested and the results can be viewed in the results section of this report. Further testing could’ve always been done in order to validate the network even further although the amount which was done so far has been deemed sufficient. An in-depth evaluation of this project can be found in the evaluation section of the report.

### 10.3 Evaluation of Personal Performance

The completion of this whole project has allowed me to acquire new sets of skills and knowledge as well as allowed me to develop and further expand the ones which I already possessed. Before undertaking this project my confidence to plan and manage projects was very little, maybe even none existent but now when all is completed I feel more then comfortable with designing and developing any other individual projects. At the start my overall knowledge and experience within the field of neural network was none existent and now I still consider myself new and a rookie but this project has sparked my interest further and I defiantly want peruse the subject further and carry on developing within this field. Overall I am very happy with my performance and progress and even the mistakes I have made have given me motivation to better myself in future endeavours.

## 11 APPENDIX A

### Part of the main class

{

public static double[][] XOR\_INPUT ={

new double[8] { 0.6974163, 1, 0.925430294, 0.301076713, 0.177980167, 0,

0.004348009, 0.087177038},

new double[8] { 0.6007378, 1, 1, 0.317689576, 0.202970596, 0.086027524, 0,

0.081713078}, new double[8] {0.741544477, 1, 1, 0.31531901, 0.374627767, 0,

0.834237537, 0},

new double[8] { 0.440140482, 1, 1, 0.220070241, 0.220070241, 0, 0,

0.110035121}, new double[8] {0.680718743, 0.854613501, 1, 0.406831477,

0.166175397, 0.080418173, 0, 0.099245696}, new double[8] {0.527019577, 0.968640584, 1, 0.386635111,

0.335241881, 0.087836596, 0.175673192, 0}, new double[8] { 0.612436867, 1, 1, 0.258883476, 0.375,

0.17539053, 0.082106781, 0 }, new double[8] {0.794201674, 1, 0.929034532, 0.381713059,

0.48780944, 0, 0.158840335, 0.238260502}, new double[8] { 0.79856847, 1, 0.95497455, 0.294989646,

0.500610405, 0, 0.11835547, 0.093558116 }, new double[8] { 1, 0.94222051, 0.889244399, 0.489949494, 0.563014581, 0,

0.306225775, 0.4 }, new double[8] { 0.898007718, 0.957186538, 1, 0.317745865,

0.39317611, 0.358837882, 0, 0.353814513},

new double[8] { 0.763469237, 1, 1, 0.307944962,

0.427678041, 0.174358147, 0.084829915, 0 },

|  |  |
| --- | --- |
| new double[8] {0.816803472, 0.959050571,  0.608685077, 0, 0.267213888, 0.804535602 }, | 1, 0.433616225, |
| new double[8] { 0.869163585, 0.917624658,  0.565043727, 0.274912911, 0, 0.255671661}, | 1, 0.151399916, |
| new double[8] { 0.906198267, 0.865396282, 1,  0.387219246, 0, 0.176050654, 0.464963225 }}; | 0.453652282, |

public static double[][] XOR\_IDEAL = {

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| new double[15] {1, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 1,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 1, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 1, |
| new double[15] { 0, 0,  1, 0, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 1, 0, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 1, 0 }, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| new double[15] { 0, 0,  0, 0, 0, 1 }}; | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| public static double[][] XOR\_test ={ | | |
| new double[8] { 0.906198267, 0.865396282, | | | | | | | 1, |  |  |
| 0.453652282, 0.387219246, 0, 0.176050654, 0.464963225 }}; | | | | | | | | |
| // 0.906198267, 0.865396282, 1, 0.453652282, 0.387219246,  0.176050654, 0.464963225 | | | | | | | | | 0, |

static void Main(string[] args)

{

FeedforwardNetwork network = new FeedforwardNetwork(); network.AddLayer(new FeedforwardLayer(8));

network.AddLayer(new FeedforwardLayer(4)); network.AddLayer(new FeedforwardLayer(15)); network.Reset();

Train train = new Backpropagation(network, XOR\_INPUT, XOR\_IDEAL,

0.2, 0.7);

int epoch = 1;

do {

train.Iteration();

Console.WriteLine("Epoch #" + epoch + " Error:" + train.Error); epoch++;

} while ((epoch < 10000) && (train.Error > 0.001));

Console.WriteLine("Neural Network Results:"); for (int i = 0; i < XOR\_IDEAL.Length; i++)

{

double[] actual = network.ComputeOutputs(XOR\_INPUT[i]); Console.WriteLine(XOR\_INPUT[i][0] + "," + XOR\_INPUT[i][1] + "," +

XOR\_INPUT[i][2] + "," + XOR\_INPUT[i][3] + "," + XOR\_INPUT[i][4] + "," +

XOR\_INPUT[i][5] + "," + XOR\_INPUT[i][6] + "," + XOR\_INPUT[i][7]

+ ", actual=" + actual[0] + "," + actual[1]+ ","+ actual[2]+ ","+ actual[2] +","+ actual[4]+ ","+ actual[5]+ ","+ actual[6]+ ","+ actual[7] +","+ actual[8]+ ","+ actual[9]+ ","+ actual[10]+ ","+ actual[11]+ ","+ actual[12] +","+ actual[13] +","+ actual[14] + ",ideal=" + XOR\_IDEAL[i][0] + XOR\_IDEAL[i][1] +

XOR\_IDEAL[i][2] + XOR\_IDEAL[i][3] + XOR\_IDEAL[i][4] + XOR\_IDEAL[i][5] + XOR\_IDEAL[i][6] + XOR\_IDEAL[i][7] + XOR\_IDEAL[i][8] + XOR\_IDEAL[i][9] +

XOR\_IDEAL[i][10] + XOR\_IDEAL[i][11] + XOR\_IDEAL[i][12] + XOR\_IDEAL[i][13] +

XOR\_IDEAL[i][14]);

// Console.ReadLine();

}

for (int i = 0; i < XOR\_test.Length; i++)

{

double[] actual = network.ComputeOutputs(XOR\_test[i]);

Console.WriteLine("ACTUAL" + actual[0] + "," + actual[1] + "," + actual[2] + "," + actual[3] + "," + actual[4] + "," + actual[5] + "," + actual[6] + "," + actual[7] + "," + actual[8] + "," + actual[8] + "," + actual[9] + "," + actual[10] + "," + actual[11] + "," + actual[12] + "," + actual[13] + "," + actual[14]);

Console.ReadLine();

}

}

}

### Sigmoid Activation function

public class ActivationSigmoid : ActivationFunction

{

public double ActivationFunction(double d)

{

return 1.0 / (1 + BoundNumbers.Exp(-1.0 \* d)); }

public double DerivativeFunction(double d)

{

return d \* (1.0 - d);

}

} }

### Backpropagation

public BackpropagationLayer(Backpropagation backpropagation,

FeedforwardLayer layer)

{

this.backpropagation = backpropagation; this.layer = layer;

int neuronCount = layer.NeuronCount;

this.error = new double[neuronCount]; this.errorDelta = new double[neuronCount];

if (layer.Next != null)

{

this.accMatrixDelta = new Matrix(layer.NeuronCount + 1, layer.Next.NeuronCount);

this.matrixDelta = new Matrix(layer.NeuronCount + 1, layer.Next.NeuronCount);

this.biasRow = layer.NeuronCount;

}

}

public void AccumulateMatrixDelta(int i1, int i2, double value)

{

this.accMatrixDelta.Add(i1, i2, value);

}

public void AccumulateThresholdDelta(int index, double value)

{

this.accMatrixDelta.Add(this.biasRow, index, value); }

public void CalcError()

{

BackpropagationLayer next = this.backpropagation

.GetBackpropagationLayer(this.layer.Next);

for (int i = 0; i < this.layer.Next.NeuronCount; i++)

{

for (int j = 0; j < this.layer.NeuronCount; j++)

{

AccumulateMatrixDelta(j, i, next.GetErrorDelta(i)

* this.layer.GetFire(j));

SetError(j, GetError(j) + this.layer.LayerMatrix[j, i]

* next.GetErrorDelta(i));

}

AccumulateThresholdDelta(i, next.GetErrorDelta(i));

}

if (this.layer.IsHidden())

{

for (int i = 0; i < this.layer.NeuronCount; i++)

{

SetErrorDelta(i, BoundNumbers.Bound(CalculateDelta(i)));

}

}

}

public void CalcError(double[] ideal)

{

for (int i = 0; i < this.layer.NeuronCount; i++)

{

SetError(i, ideal[i] - this.layer.GetFire(i));

SetErrorDelta(i, BoundNumbers.Bound(CalculateDelta(i)));

}

}

private double CalculateDelta(int i)

{

return GetError(i) \*

this.layer.LayerActivationFunction.DerivativeFunction(this.layer.GetFire(i));

}

public void ClearError()

{

for (int i = 0; i < this.layer.NeuronCount; i++)

{

this.error[i] = 0;

}

}

public FeedforwardNetwork Network

{ get {

return this.network;

}

}

public double Error

{ get {

return this.error;

}

}

private double error;

private double learnRate;

private double momentum;

private FeedforwardNetwork network;

private IDictionary<FeedforwardLayer, BackpropagationLayer> layerMap = new

Dictionary<FeedforwardLayer, BackpropagationLayer>();

private double[][] input;

private double[][] ideal;

public Backpropagation(FeedforwardNetwork network, double[][] input, double[][] ideal, double learnRate, double momentum)

{ this.network = network; this.learnRate = learnRate; this.momentum = momentum; this.input = input; this.ideal = ideal;

foreach (FeedforwardLayer layer in network.Layers) {

BackpropagationLayer bpl = new BackpropagationLayer(this, layer);

this.layerMap.Add(layer, bpl);

}

}

**12**

**A**

**PPENDIX**

**B**

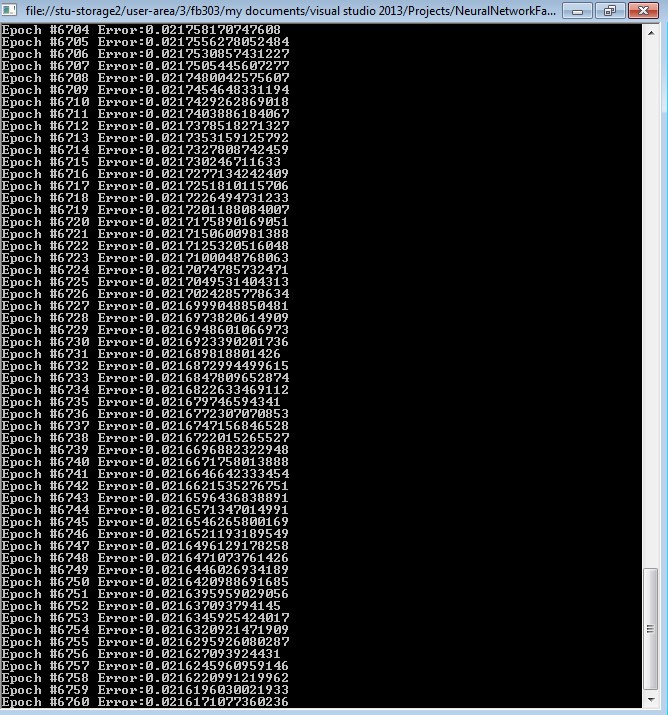
One of the print screens above present the network

running and performing iterations while the seconed

print screen show the networks o

utput as well as the

testing values for the last neuron.



### 13 APPENDIX C

**13.1 The data sets used for training prototype 2 and 3.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.6974  163 | 1 | 0.9254  30294 | 0.3010  76713 | 0.1779  80167 | 0 | 0.0043  48009 | 0.0871  77038 |
| 0.6007  378 | 1 | 1 | 0.3176  89576 | 0.2029  70596 | 0.0860  27524 | 0 | 0.0817  13078 |
| 0.7415  44477 | 1 | 1 | 0.3153  1901 | 0.3746  27767 | 0 | 0.8342  37537 | 0 |
| 0.4401  40482 | 1 | 1 | 0.2200  70241 | 0.2200  70241 | 0 | 0 | 0.1100  35121 |
| 0.6807  18743 | 0.854613501 | 1 | 0.4068  31477 | 0.1661  75397 | 0.0804  18173 | 0 | 0.0992  45696 |
| 0.5270  19577 | 0.968640584 | 1 | 0.3866  35111 | 0.3352  41881 | 0.0878  36596 | 0.1756  73192 | 0 |
| 0.6124  36867 | 1 | 1 | 0.2588  83476 | 0.375 | 0.1753  9053 | 0.0821  06781 | 0 |
| 0.7942  01674 | 1 | 0.9290  34532 | 0.3817  13059 | 0.4878  0944 | 0 | 0.1588  40335 | 238260  502 |
| 0.7985  6847 | 1 | 0.9549  7455 | 0.2949  89646 | 0.5006  10405 | 0 | 0.1183  5547 | 0.0935  58116 |
| 1 | 0.94222051 | 0.8892  44399 | 0.4899  49494 | 0.5630  14581 | 0 | 0.3062  25775 | 0.4 |
| 0.8980  07718 | 0.957186538 | 1 | 0.3177  45865 | 0.3931  7611 | 0.3588  37882 | 0 | 0.3538  14513 |
| 0.7634  69237 | 1 | 1 | 0.3079  44962 | 0.4276  78041 | 0.1743  58147 | 0.0848  29915 | 0 |
| 0.8168  03472 | 0.959050571 | 1 | 1 | 0.4336  16225 | 0.6086  85077 | 0.2672  13888 | 0.8045  35602 |
| 0.8691  63585 | 0.917624658 | 1 | 0.1513  99916 | 0.5650  43727 | 0.2749  12911 | 0 | 0.2556  71661 |
| 0.9061  98267 | 0.865396282 | 1 | 0.4536  52282 | 0.3872  19246 | 0 | 0.1760  50654 | 0.4649  63225 |

### 14 APPENDIX D

#### 14.1 Testing for prototype 1

|  |  |  |
| --- | --- | --- |
| Input | Idea Output | Actual Output |
| 0,0 | 0 | 0.00676012877274516 |
| 1,0 | 1 | 0.993341551788032 |
| 0,1 | 1 | 0.993542275201883 |
| 1,1 | 0 | 0.00770856851540549 |

#### 14.2 Testing for prototype 2

|  |  |  |
| --- | --- | --- |
| Input | Ideal Output | Actual Output |
| 0.6974163,1, 0.925430294,  0.301076713, 0.177980167, 0,  0.004348009, 0.087177038 | 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 | 0.91827467364847 |

|  |  |  |
| --- | --- | --- |
| 0.6007378, 1, 1, 0.317689576,  0.202970596, 0.086027524, 0,  0.081713078 | 0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 | 1.39475873526256 |
| 0.741544477, 1, 1,  0.31531901, 0.374627767, 0,  0.834237537, 0 | 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0 | 0.81274654637627 |
| 0.440140482, 1, 1,  0.220070241, 0.220070241, 0,  0, 0.110035121 | 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 | 0.99434773647365 |
| 0.680718743, 0.854613501, 1,  0.406831477, 0.166175397,  0.080418173, 0, 0.099245696 | 0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 | 1.194572463451562 |
| 0.527019577, 0.968640584, 1,  0.386635111, 0.335241881,  0.087836596, 0.175673192, 0 | 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 | 0.934367566475636 |
| 0.612436867, 1, 1,  0.258883476, 0.375,  0.17539053, 0.082106781, 0 | 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0 | 1.223243545549858 |
| 0.794201674, 1, 0.929034532,  0.381713059, 0.48780944, 0,  0.158840335, 0.238260502 | 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 | 0.874587673672878 |
| 0.79856847, 1, 0.95497455,  0.294989646, 0.500610405, 0,  0.11835547, 0.093558116 | 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 | 1.176765432455667 |
| 1, 0.94222051, 0.889244399,  0.489949494, 0.563014581, 0,  0.306225775, 0.4 | 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 | 1.434465435575455 |
| 0.898007718, 0.957186538, 1,  0.317745865, 0.39317611,  0.358837882, 0, 0.353814513 | 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0 | 0.935676487847743 |
| 0.763469237, 1, 1,  0.307944962, 0.427678041,  0.174358147, 0.084829915, 0 | 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0 | 0.895684786736743 |
| 0.816803472, 0.959050571, 1,  0.433616225, 0.608685077, 0,  0.267213888, 0.804535602 | 0,0,0,0,0,0,0,0,0,0,0,0,1,0,0 | 1.11884754888874836 |
| 0.869163585, 0.917624658, 1,  0.151399916,  0.565043727,0.274912911, 0,  0.255671661 | 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0 | 0.868784372463778 |
| 0.906198267, 0.865396282, 1,  0.453652282, 0.387219246, 0,  0.176050654, 0.464963225 | 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 | 1.599994958654787 |

#### 14.3 Test nr.1 for prototype 3

|  |  |  |
| --- | --- | --- |
| Input | Ideal Output | Actual Output |
| 0.6974163,1, 0.925430294,  0.301076713, 0.177980167, 0,  0.004348009, 0.087177038 | 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 | 0.93438743878443 |
| 0.6007378, 1, 1, 0.317689576,  0.202970596, 0.086027524, 0,  0.081713078 | 0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 | 0.92834765748674 |
| 0.741544477, 1, 1,  0.31531901, 0.374627767, 0,  0.834237537, 0 | 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0 | 0.99765988435763 |
| 0.440140482, 1, 1,  0.220070241, 0.220070241, 0,  0, 0.110035121 | 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 | 0.92374677467655 |
| 0.680718743, 0.854613501, 1,  0.406831477, 0.166175397,  0.080418173, 0, 0.099245696 | 0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 | 0.91284736574576 |
| 0.527019577, 0.968640584, 1,  0.386635111, 0.335241881,  0.087836596, 0.175673192, 0 | 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 | 0.99945894859848 |
| 0.612436867, 1, 1,  0.258883476, 0.375,  0.17539053, 0.082106781, 0 | 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0 | 0.95243545549858 |
| 0.794201674, 1, 0.929034532,  0.381713059, 0.48780944, 0,  0.158840335, 0.238260502 | 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 | 0.97889965437677 |
| 0.79856847, 1, 0.95497455,  0.294989646, 0.500610405, 0,  0.11835547, 0.093558116 | 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 | 0.98877674677436 |
| 1, 0.94222051, 0.889244399,  0.489949494, 0.563014581, 0,  0.306225775, 0.4 | 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 | 0.92454575874748 |
| 0.898007718, 0.957186538, 1,  0.3177  45865, 0.39317611,  0.358837882, 0, 0.353814513 | 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0 | 0.99237383767346 |
| 0.763469237, 1, 1,  0.307944962, 0.427678041,  0.174358147, 0.084829915, 0 | 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0 | 0.97895476327633 |
| 0.816803472, 0.959050571, 1,  0.433616225, 0.608685077, 0,  0.267213888, 0.804535602 | 0,0,0,0,0,0,0,0,0,0,0,0,1,0,0 | 0.95736765543673 |
| 0.869163585, 0.917624658, 1,  0.151399916,  0.565043727,0.274912911, 0,  0.255671661 | 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0 | 0.912743874384773 |
| 0.906198267, 0.865396282, 1,  0.453652282, 0.387219246, 0,  0.176050654, 0.464963225 | 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 | 0.918377489839743 |

#### 14.4 Test nr 2

Player 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.7974162, 1,  0.825430294,  0.301076713,  0.177980167, 0  00.3348009,  0.077177038 | 0.6874163, 1,  0.925430294,  0.201076713,  0.177980167, 0,  0.004348009,  0.077177038 | 0.6874164, 1,  0.825430294,  0.201076713,  0.177980167, 0,  0.005348009,  0.067177038 | 0.6874164, 1,  0.825430214,  0.201076713,  0.077980167, 0,  0.005348009,  0.067177038 | 0.5864164, 1,  0.825430214,  0.101076713,  0.077980167, 0,  0.004348009,  0.067177038 |
| 0.995369653975  295 | 0.986551350155  056 | 0.919089683165  957 | 0.999350302667  576 | 0.997346319622  209 |

Within the table above, all of the testing results from player one can be seen. This clearly shows show that the network has been successfully trained for this particular face. Although some the results were quite close to the edge they do fit within the stated boundaries.

Player 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.7007377, 1, 1,  0.117689575,  0.202970596,  0.096027524, 0,  0.091713078 | 0.5007277, 1, 1,  0.117689575,  0.202960596,  0.086027524, 0,  0.092713079 | 0.4007277, 0, 1,  03117689565,  0.002840596,  0.096027524, 0,  0.082713077 | 0.5007276, 1, 1, 04117689565, 0.202960596,  0.096026524, 0,  0.072713079 | 0.6006276,  1, 1,  03116689565,  0.212960596,  0.086125524,  0,  0.0527130069 |
| 0.9983854936523  54 | 0.9099056568144  74 | 0.1598545698595  53 | 0.987821328878  57 | 0.9377066346  7 |

Although the majority of results within this table fit within the boundaries, one does not. The result misses by a quite large amount. This could be an argument that too many iterations were used during the training. This is because only one of the images missed the targeted boundaries, if the case was reversed the issue would most likely be that not enough iterations were performed.

Player 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.641544575,  1, | 1, | 0.631554565,  1, | 1, | 0.511454565,1,  1, 0.01431900, | 0.651653564,1, 1, 0.03431990, | 0.662665565,1, 1,  0.02331870, |
| 0.21531801,  0.474527766, 0,  0.734237537, 0 | | 0.11431901,  0.364527766,  0,  0.534237533, 0 | | 0.464527763,  0,  0.551237535, 0 | 0.455526762,  0,0.651237535, 0 | 0.436527766,  0,0.611334431, 0 |
| 0.993201681223  091 | | 0.98081282184  602 | | 0.09671231183  128 | 0.9109111256783  199 | 0.931311233568241  823 |

This table shows another successfully trained a neuron. The results from five different photos of one person are shown above and that all of them fitting inside the boundaries set within the design section.

Player 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.341240472, 1, 1, 0.130060242, 0.121060232, 0, 0, 0.103035341 | 0.241250432, 1, 1, 0.224040241, 0.221060234, 0, 0, 0.211535344 | 0.351251431, 1, 1, 0.124141243, 0.221260254, 0, 0, 0.203655378 | 0.451221471, 1, 1, 0.231046221, 0.221230264, 0, 0, 0.112535546 | 0.332342371, 1, 1, 0.112360242, 0.121261331, 0, 0, 0.123034549 |
| 0.977872460795  732 | 0.923272411875  465 | 0.936473456956  734 | 0.993953456723  452 | 0.905454758647  396 |

Just like the neuron above, this training was also a success. There the fourth photo within this neuron was quite close to overstepping the boundaries but nonetheless this is another successful iteration.

Player 5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.771618732,  0.813614  321, 1,  0.406431  238,  0.156355  161,  0.070418  181, 0,  0.099245  696 | 0.630    401,    466,    368,    162,    695 | 718723,  0.754612  1,  0.306831  0.166775  0.070417  0,  0.098245 | 0.681    13,    76,    96,    62,    95 | 618732,  0.9556135  1,  0.5168314  0.2751753  0.0994182  0,  0.0892455 | 0.782    4,    8,    5,    2,    4 | 718754,  0.74461353  1,  0.52583146  0.26517538  0.07231726 0,  0.08724563 | 2,    5,    6,    2,    6 | 0.487718732,  0.76552351  0,  0.31583123  0.15616528  0.09151816  1,  0.09813569 |
| 0.96681993059  646 | 0.93282395768  305 | | 0.913486749389  726 | | 0.9923583675839  271 | | 0.8723585738498  471 | |

All of but one iteration were tested successfully therefore is can be stated that the overall nenuron proved to be a success.

Player 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.416118576,  0.858640  534, 1,  0.349763  5134,  0.442241  8321,  0.077936  466,  0.265674  582, 0 | 0.628019567,  0.968675  584, 1,  0.286633  434,  0.235241  791,  0.088837  586,  0.164673  347, 0 | 0.437    73,    12,    21,    95,    92, | 128566,  0.8786405  1,  0.4775352  0.4462417  0.0779364  0.1745731  0 | 0.516    3,    2,    2,    5,    1, | 111687,  0.86753158  1,  0.27763521  0.34434188  0.09793669  0.26667319  0 | 3,    1,    2,    7,    1, | 0.536019536,  0.97764048  1,  0.48564522  0.23424197  0.08793669  0.16677318  0 |
| 0.92486929765  3 | 0.99895759763  784 | 0.987774652837  465 | | 0.9037434585736  784 | | 0.9566436784756  248 | |

This is another successfully trained neuron by the network as all of the iterations performed fitted within the required boundaries.

Player 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.422      76,    4,    82, | 436767,  1, 1,  0.3598844 0.675,  0.1853105  0.0921059  0 | 0.513436856,  1, 1,  0.1489336  6, 0.284,  0.1663915  3,  0.0721167  81, 0 | 0.612436867,  1, 1,  0.1579733  76, 0.277,  0.2843905  3,  0.0822057  91, 0 | 0.522567867,  1, 1,  0.2487744  65, 0.465,  0.1864905  4,  0.0711077  82, 0 | 0.613435766,  0, 1,  0.3597834  76, 0.264,  0.1644804  3,  0.0921067  71, 0 |
| 0.992554958674  623 | | 0.912954756382  734 | 0.933955746586  754 | 0.993433647563  647 | 0.893456739294  783 |

Similary to some of the neurons before one iterationfor this neorn did not fit between the required boundry. Nevertheless majority of iterations performed came back with positive results therefore it can be stated that this neuron has been successfully trained.

Player 8

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31,    58,    5,    36,    12 | 0.884311673,  1,  0.8281345  0.2717131  0.5978194  0,  0.2688413  0.1292615 | 33,    69,    3,    44,    13 | 0.894101674,  1,  0.9281245  0.4927130  0.3778184  0,  0.1479402  0.3392605 | 2,    8,    ,    4,    1 | 0.783301573,  1,  0.81812442  0.47281415  0.59891956  0,  0.17995043  0.13835051 | 41,    28,    4,    94,    01 | 0.695212563,  1,  0.8190446  0.2806131  0.3977093  0,  0.2479412  0.3472623 | 74,    58,    4,    25,    01 | 0.684322685,  1,  0.9291442  0.2936230  0.5969185  0,  0.1678414  0.1491505 |
| 0.923455674899  387 | | 0.972554576577  382 | | 0.9122346777634  374 | | 0.931244675641  431 | | 0.973657616316  489 | |

Every image which was tested for this neuron was fitted perfectly within the boundaries set.

Player 9

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.89755846, 1,  0.84487446,  0.195888645,  0.412620416,  0,  0.12826548,  0.113558125 | 0.69846142    5,    23,    35,    5,    28 | , 1,  0.8159471  0.3949835  0.6106162  0,  0.1123523  0.0936956 | 2,    45,    28,    36,    495 | 0.713434857,  1,  0.8385638  0.2923917  0.5184938  0,  0.1294857  0.1274857 | 0.798    55,    646,    405,    47,    116 | 56847, 1,  0.954974  0.294989  0.500610  0,  0.118355  0.093558 | 0.79856847    5,    46,    05,    7,    16 | , 1,  0.9549745  0.2949896  0.5006104  0,  0.1183554  0.0935581 |
| 0.992384756283746  453 | 0.912847611290  394 | | 0.938495756554  535 | | 0.93284657463  726 | | 0.993495895869  506 | |

Similar to the neuron before the outputs above are a result of another successful training.

Player 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1, 0.97685673,  0.9394875678,  0.499408328,  0.5928938475, 0,  0.489583782, 0.5 | 1, 0.89389473,  0.796874767,  0.374856781,  0.485968788, 0,  0.289476847, 0.3 | 1, 0.998475768,  0.94859685711,  0.58675875142,  0.6947991312, 0,  0.475684321, 0.4 | 1, 0.8664323,  0.0941324132,  0.576543421,  0.425237876, 0,  0.496786721, 0.3 | 1, 0.9764765,  0.976476589,  0.486542812,  0.656788543, 1,  0.309876569, 0.4 |
| 0.9364758918309 | 0.91293847621 | 0.96756372134 | 0.897563736252 | 0.82758476359 |

This neuron was very close to failing although the majority of iterations were positive. Therefore the testing of this neuron falls in the success column. As two of the iterations have failed further investigation should be done.

Player 11

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.    51,    29,    7,    13,    69 | 899475864,  0.9898936  1,  0.3879784  0.4123184  0.3898932  0,  0.3456365 | 27,    2,    2,    9,    9 | 0.878957634,  0.9323389  1,  0.4355231  0.3994746  0.3749883  0,  0.3132456 | 12,    37,    2,    56,    35 | 0.823948595,  0.9247847  1,  0.2989478  0.4298236  0.3434657  0,  0.3435675 | 45,    98,    1,    72,    54 | 0.798979768,  0.9123243  1,  0.2898378  0.4323243  0.3678475  0,  0.3234324 | 45,    54,    8,    34,    34 | 0.786987847,  0.9453465  1,  0.3234235  0.4236678  0.3754454  0,  0.3432364 |
| 0.912647567384  2 | | 0.943673627674  8 | | 0.946377463627  6 | | 0.994399374737  6 | | 0.986764356545  4 | |

This neuron has been successfully trained with not negative results.

Player 12

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.754334234,  1, 1, | | 0.7786747678,  1, 1, | | 0.7543456337,  1, 1, | | 0.65775689, 1,  1, | | 0.76567454, 1,  1, | |
| 98,    38,    32,    99, | 0.2938584  0.4123847  0.1446674  0.0784759  0 | 57,    88,    74,    21, | 0.3182746  0.4398787  0.1857487  0.0987452  0 | 54,    7,    47,    64, | 0.35567654  0.44365567  0.19678957  0.09984575  0 | 6,    6,    5,    6, | 0.3128473  0.3998978  0.2134345  0.0832454  0 | 6,    5,    7,    6, | 0.3054646  0.4124343  0.1654576  0.0937483  0 |
| 0.904548576763  6 | | 0.943546567657  4 | | 0.8954545653234  4 | | 1.249357467553  3 | | 0.954356543248  4 | |

Another neuron has failed two iterations therefore although the training of this neuron is deemed a success a investigation has to be certinaly carried out in order to fin what’s at fault. Player 13

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.8    63,    87,    76,    47,    87 | 048373627,  0.9434565  1,  0.3995857  0.5968673  0,  0.2596888  0.8124598 | 67,    65,    65,    78,    76 | 0.823445466,  0.9567656  1,  0.4978767  0.6233454  0,  0.3456576  0.8456565 | 44,    76,    67,    6,    36 | 0.7958457664,  0.9767545  1,  0.4243565  0.6122455  0,  013455657  0.7984577 | 23,    83,    23,    35,    54 | 0.832343545,  0.9654343  1,  0.4399897  0.7234202  0,  0.2967444  0.8123546 | 65,    53,    56,    73,    34 | 0.795484347,  0.9436834  1,  0.4124356  0.6234567  0,  0.2754535  0.9224312 |
| 0.945434275548  3 | | 0.998765434343  2 | | 0.997656999979  7 | | 0.954587986544  7 | | 0.874556744345  6 | |

One Iterations has produced a faulty result, but the training of the network is still deemed a success as the majority produced the expected output.

One

Player 14

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.8987985875,  0.9256645 | | 0.8750968958,  0.8985478 | | 0.91234456543,  0.914657546 | 0.8867568745,  0.913556 | 0.8535568796,  0.9345653 | |
| 66,    87,    78,    78,    98 | 1,  0.1656767  0.5948596  0.1987685  0,  0.2498878 | 77,    88,    34,    34,    09 | 1,  0.1499457  0.5124346  0.2856754  0,  0.2657689 | , 1,  0.149857873  ,  0.623434543  ,  0.264556776  , 0,  0.247543654 | 54, 1,  0.156564  5,  0.554754  45,  0.313234  3, 0,  0.243645 | 34,    73,    09,    4,    5 | 1,  0.1694958  0.5784578  0.2589873  0,  0.2656754 |
| 0.924545675677  6 | | 0.923464323435  6 | | 0.91345575466756  5 | 0.89989548487  8 | 0.944545757878  7 | |

Similary to the neuron before only one of the neurons has faild therefore this falls within the success column. Player 15

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 34,    84,    89,    87,    46 | 0.923435445,  0.8765645  1,  0.4466877  0.3998547  0,  0.1698987  0.5143435 | 78,    86,    53,    34,    76 | 0.919898498,  0.8546576  1,  0.4465676  0.3756434  0,  0.1811212  0.4536678 | 99,    43,    54,    54,    76 | 0.899756655,  0.8598948  1,  0.4645455  0.3922334  0,  0.1865656  0.4434668 | 0.91234355    43,    57,    45,    47,    54 | ,  0.8798765  1,  0.4235456  0.3967675  0,  0.1694988  0.4967556 | 0.93557678    37    56,    74,    73,    34 | ,  0.8985484  1,  0.4545327  0.3965948  0,  0.1994573  0.4756543 |
| 0.914636567684  6 | | 0.911345324457  8 | | 0.946865548656  4 | | 0.966897689676  7 | | 1.004667785455  6 | |

A large majority of iterations returned positive results even if one of them produced a result which did not produced he required result the neuron can still be deemed a success.

#### 14.5 Test nr.3

|  |  |  |
| --- | --- | --- |
| Input | Ideal Output | Actual Output |
| 0.3546567533,0, 0.934738476,  0.545465657, 1, 0, 0.12243545,  0.0768764546 | 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 | 0.73948938490454 |
| 0.434657655, 0.345767, 0,  0.8347837483, 0.988477473,  1, 0, 0.07654667 | 0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 | 0.549689578958994 |
| 0.435564344, 0, 1, 1,  0.2345465565, 0,  0.576876755, 0 | 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0 | 0.754767378267478 |
| 0.986374673, 1,  0.9548578572,  0.234545754588, 0.134565, 1,  0, 0.3545768787 | 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 | 0.8374679942388238 |
| 1, 1, 0.648587485,  0.57654557, 0.2934845748, 1,  0, 0.9586495080 | 0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 | 0.2393589458478954 |
| 0.25469856656,  0.94858764958, 0, 1,  0.1235465, 0.34556734565, 1,  0 | 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 | 0.3465676754647655 |
| 0.4655784, 0, 0.3494584754,  0.9875473, 0.2356578970,  0.1724785784, 0.13458978, 1 | 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0 | 0.923446576787 |
| 1, 0.9787654876,  0.847588498, 0.03398547059,  0.876347644, 1,  0.3445845784, 0.194874574 | 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 | 0.913549659694994 |
| 0.498780383, 0, 0.85789733,  1, 0.084574757, 0,  0.2445575t45, 0.9734834675 | 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 | 0.745767393878387845 |
| 0.73476434, 0.12454467674,  0..98578476587, 1,  0.03894347584, 0,  0.1345576574, 0.93785748857 | 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 | 0.385948596895894958 |
|  |  |  |
| 1, 0.08745375745, 0,  0.97857549, 0.74989438949,  1, 0, 0.46574345676 | 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0 | 0.924938594584989434 |
| 0.94849565667,  0.93948845746, 1,  0.245465776, 0.6984978475,  1, 0.3656767876, 1 | 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0 | 0.885783478373245345 |
| 0.948375487, 1,  0.34767546754,  0.0384738545, 0.243456675,  0, 0.034546557,  0.23454575754 | 0,0,0,0,0,0,0,0,0,0,0,0,1,0,0 | 0.794857834594869 |
| 0.038475454878, 1, 1,  0.346567675, 0.2334545767,  0.0349754, 0, 1 | 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0 | 0.694853975948979 |
| 0.0457346534, 1, 1,  0.024565734, 0.234546557, 0,  1, 0.465599685990 | 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 | 0.789843537459793 |

## 15 APPENDIX E

### 15.1 Images used for training



### 15.2 Images for testing

Bolateli





Buffon



Pazini



Szczesny



Obraniak



Lewandowski



Bendtner



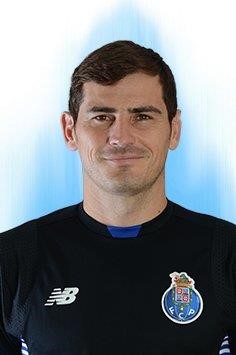
Villa



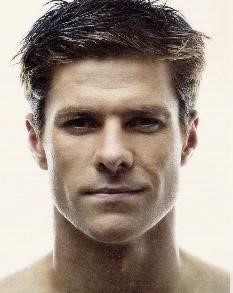
Kjaer



Casillas



Alonzo



Ivan



Lyka



Srna



Remy



### 15.3 Images used within the final testing

Naymar



David



Henry



Rooney



Edwin



Owen



Gerrard



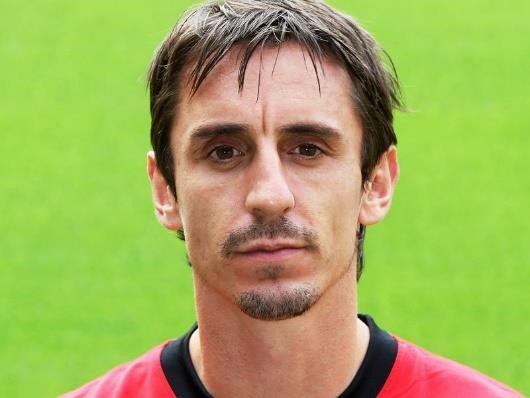
Gea



Ronaldo



Gary



George



Paul



Emile



Hazard



Jamie



### 16 BIBLIOGRAPHY

Van Der Smagt, P., 1990. A Comparative Study of Neural Network Algorithms applied to Optical Character Recognition. In 1990 International Conference on Industrial and Engineering Applications of Artifical Intellengence and Expect Systems, Volume N/A. New York, pp. 1037-1044.

1. Sirovich and M. Kirby, “Low-Dimensional procedure for the characterisation of human faces,” J. Optical Soc. of Am., vol. 4, pp. 519-524, 1987.

Link: http://www.face-rec.org/interesting-papers/General/ld.pdf

[Accessed November 2015]

1. Kirby and L. Sirovich, “Application of the Karhunen- Loève procedure for the characterisation of human faces,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 831-835, Dec. 1990.

Link<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.411.6307&rep=rep1&type=pdf>

[Accessed October 2015]

M. Turk and A. Pentland, “Eigenfaces for recognition,” J. Cognitive Neuroscience, vol. 3, pp. 71-86, 1991.

Linkhttp://s3.amazonaws.com/academia.edu.documents/30894770/jcn.pdf?AWSAccessKeyId=

AKIAJ56TQJRTWSMTNPEA&Expires=1460970881&Signature=Hi8kKAEN3XPqeGAOm6L

PMkZTcMA%3D&response-content-

disposition=inline%3B%20filename%3DEigenfaces\_for\_Recognition.pdf

[Accessed January 2016]

L. ZHAO AND Y.H. YANG, “THEORETICAL ANALYSIS OF ILLUMINATION IN PCA- based vision systems,” Pattern Recognition, vol. 32, pp. 547-564,

1999. Link<http://www.sciencedirect.com/science/article/pii/S0031320398001198>

[Accessed November 2015]

A. Pentland, B. Moghaddam, and T. Starner, “View-Based and modular eigenspaces for face recognition,” Proc. IEEE CS Conf.

Computer Vision and Pattern Recognition, pp. 84-91, 1994.

1999. Link<http://vismod.media.mit.edu/pub/facereco/papers/TR-245.pdf>

[Accessed October 2015]

Rogers, S. K. & Kabrisky, M., 1991. An Introduction to Biological and Artificial Neural Networks for Pattern Recognition. pp. 44-50 1st ed. Washington USA: The International Society of Optical Engineering.

T.J. STONHAM, “PRACTICAL FACE RECOGNITION AND VERIFICATION WITH WISARD,” ASPECTS OF FACE PROCESSING, PP. 426-441, 1984.

Link http://link.springer.com/chapter/10.1007/978-94-009-4420-6\_44#page-1 [Accessed October 2015]

J. Weng, J.S. Huang, and N. Ahuja, “Learning recognition and segmentation of 3D objects from 2D images,” Proc. IEEE Int'l Conf. Computer Vision, pp. 121-128, 1993.

Link http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=378228

[Accessed January 2016]

OpenCV, 2012. Open Source Computer Vision.

Link http://opencv.org/

[Accessed November 2015].

NeuroSolutions, 2012. What is a Neural Network?.

Link http://www.neurosolutions.com/products/ns/whatisNN.html [ Accessed October 2015].