# 

# 

# 

# 

# 

# 

# 

# 

# 

# **ECE 457B Project: Facial Expression Recognition System**

Jason Ku 20382089 [j3ku@uwaterloo.ca](mailto:j3ku@uwaterloo.ca)

Caleb Yu 20378827 [cycyu@uwaterloo.ca](mailto:cycyu@uwaterloo.ca)

Corey Kirschbaum 20333526 [cakirsch@uwaterloo.ca](mailto:cakirsch@uwaterloo.ca)

April 6, 2015

# 

Contents

[Abstract 1](#_Toc416036196)

[Introduction 2](#_Toc416036197)

[Why Computational Intelligence? 2](#_Toc416036198)

[Theoretical Background 3](#_Toc416036199)

[Artificial Neural Networks (ANN) 3](#_Toc416036200)

[Relevant research 4](#_Toc416036201)

[Conclusions from References 4](#_Toc416036202)

[Implementation 6](#_Toc416036203)

[Results 10](#_Toc416036204)

[Varying Number of Neurons 10](#_Toc416036205)

[Varying Number of Hidden Layers 11](#_Toc416036206)

[Cumulative Error and Classification Results 12](#_Toc416036207)

[Varying Input Size 13](#_Toc416036208)

[Real Time Performance and Memory Size 15](#_Toc416036210)

[Conclusion 17](#_Toc416036211)

[References 18](#_Toc416036212)

# 

# Abstract

This report explores the use of neural networks for application in the field of human-machine interfacing; specifically, detecting facial kinesic (body language) cues during human communication for the purpose of increasing the accuracy and depth of interpretation of the human’s intent. To this end, this study looks at the effects of removing certain facial features from inputs to the neural network. This restriction is placed since a person’s mouth will move and may therefore be unreliable to use as an input while a person is talking. How accurate are neural network based facial expression recognition systems when this is removed? Can sufficient accuracy still be achieved with the remaining data? This study seeks to answer this question.

The first step is to create a neural network to perform basic facial expression detection. This system is implemented as a MATLAB script and is compared to the native neural network functions in MATLAB. The inputs are provided by the Cohn-Kanade AU-Coded Expression Database [1] as the x and y positions of facial landmarks. These are then processed through the neural network to produce one of four outputs. The facial expression outputs are happy, sad, angry, and surprised.

The neural network is tested with a varying number of nodes and number of hidden layers in order to determine the parameters which lead to the best accuracy. A single layer, 50 neuron classification neural network is empirically determined to be the most accurate with an accuracy of 91 %.

With the basic neural network complete, testing with restricted data is done. The facial features are broken into five categories: jaw, mouth, eyebrows, eyes, and nose. The system is then tested with various combinations of categories, including one with all features removed except the mouth. This testing reveals that removing the mouth leads to a reduced accuracy of 70.9%. This outcome may be due to the importance of the mouth for identification or simply a result of the fact that the mouth section contains the largest set of data. Still the results demonstrate that some reliable results can be obtained even without the mouth data.

Finally, performance tests are done to see the appropriateness of the system for real time applications. If the system is to be used when a user to speaking to a device, low latency and a low memory footprint are desired. Testing demonstrates that the system is able to classify a single input in approximately 0.04 ms. versus the 6 ms taken by the native MATLAB functions. The total memory size required at worst is around 60 KB which is very small. Both of these system characteristics mean that the algorithm is appropriate for use in handheld devices.

# 

# Introduction

Humans interaction has a great depth of complexity that goes beyond spoken words. These non-verbal forms of communication can be broken down into multiple categories including: body language (kinesics), voice (paralanguage), touch (haptics), and distance (proxemics) [1]. To ignore these cues is to miss out on the full understanding of what an individual is trying to express. This is why modes of communication should as texting or phone calls can sometimes feel stilted and unnatural.

Human-machine interaction has come a long way in recent years with popular applications such as Siri [2] attempting to parse and understand normal human speech patterns to moderate success. There has also been some work with determining caller emotion for call centre workers [3] by using paralinguistics. Many of these applications are made possible by the advances in computational intelligence.

This project will explore the area of kinesics or body language, specifically facial expressions. Facial expressions contain information regarding what an individual is feeling and can be used to gauge their reactions. Currently human-machine interactions feel artificial mainly due to AI appearing ‘cold’ and emotionless. Giving AI the ability to detect and react to a user’s mood would be a great benefit, helping individuals to be more comfortable with using AI. Unfortunately, recognizing moods from facial expressions can be difficult even for humans. Individuals can also learn to control their expressions in order to hide their true emotions. As mentioned previously, there is a large body of research in kinetics in order to read people’s moods. This expert knowledge can be applied using computational intelligence techniques in order to allow AIs to access this non-verbal information.

For the application of human-machine interfacing, the human will also be communicating to the AI verbally. This means that determining facial expression during communication will require that data regarding the mouth be ignored since the mouth will be moving and therefore unlikely to be representative of the human emotion. Normally, mouth position is heavily related to some of the emotions. For example, frowning is linked to sadness or anger. Exploring the effects of removing the reliance on this data will be a focus of this study.

### Why Computational Intelligence?

Expression recognition is fundamentally a classification problem. The system will need to process visual data and then process the data into classes of facial expressions. Neural networks, a popular application of computational intelligence, are very compatible for problems related to classification. By providing facial data inputs linked to facial expression outputs, the network can be trained to identify new data. Simply setting up an algorithm would be very difficult since there is a large variance with facial expressions from person to person. Such an algorithm would also be solely dependent on the human intuition and as a result subtle cues that are noticeable in the data, but not intuitive to humans may be overlooked. This study will use the multi-layer perceptron due to its versatility.

# Theoretical Background

### Artificial Neural Networks (ANN)

This section will outline the theoretical concepts behind artificial neural networks. It will focus on the techniques and algorithms applied in this study.

#### Perceptron and Multilayer Perceptron

The perceptron is the fundamental building block of the ANN. It is composed of any number of inputs, which are then individually weighted. The weighted inputs are then summed and passed through a squashing function to produce a binary output. Alone, it is only a linear classifier. As such non-linearly-separable data sets cannot be classified accurately. This weakness is overcome by combining multiple perceptrons into a Multi-Layer Perception (MLP). The MLP, as the name suggests, has multiple layers of multiple perceptrons. There is an input layer and output layer. The remaining layers are called hidden layers. The addition of these layers allows the ANN to classify complex sets of data.

Training

The ANN must be trained before it is able to classify facial expressions. This training is achieved by providing input data and corresponding target outputs. For example, in this study, training is done by providing a complete set of facial data representing the various output facial expressions. The data is then processed by the ANN to produce an output. That output is then compared to the actual target output and the error is calculated. This error is then processed backwards through the network and used to improve the accuracy of the ANN. This process uses the backpropagation algorithm.

When training the system, there are two options - online and offline learning. With online learning, new data is streamed in and the system is updated after each new input. However, when implemented, the system was very slow to converge and did not provide useful results. As well, since this process requires a teacher identifying the emotion before it enters the system, this method was not explored in depth in this study. One possible related application is for use in personal calibration for smart phones or other personal devices. This study will instead use offline training. This learning method relies on a large static data set which is used to train the system. Part of the data will be used for training while a small remainder will be used for testing and validating the accuracy of the network. Since facial expression recognition would require many data samples from various persons in order to achieve greater accuracy for general purpose use, this method is more appropriate.

#### Gradient Descent and Momentum

The training algorithm typically uses a gradient descent in order to minimize the error. The gradient points towards the higher values and so inversely, following the negative gradient should lead to lower values. One issue with this technique is local minima. It is possible that the gradient leads to the lowest point in an area which is not the absolute lowest point for the entire system. One way to combat this issue is to use momentum. A small percentage of the previous gradient is reused in order to help the process escape the local minima. Using momentum is not a perfect solution, but its effects will be explored in this study.

### Relevant research

There has already been some extensive research in applying computational intelligence systems for the purposes for facial recognition. These reports were explored and analyzed in order to provide direction for the study.

#### Emotion Detection Algorithm Using Frontal Face Image

Moon Hwan Kim, Young Hoon Joo, and Jin Bae Park [4]

This paper, written in 2005, applied fuzzy logic systems to facial expression recognition. The algorithm involved three stages, the image processing, facial feature extraction, and finally expression identification. For the image processing and facial feature extraction, the face was broken into three sections, the eye, mouth, and auxiliary section. Each section was analyzed for metrics such as openness of the eye or distance between the nose and mouth. These metrics were then processed into a fuzzy classiﬁer in order to identify five emotions: happy, sad, angry, disgust, and surprise. The algorithm had a classification accuracy of 89.5%.

#### Automatic Facial Feature Extraction and Expression Recognition Based on Neural Network

S.P.Khandait, Dr. R.C.Thool, and P.D.Khandait [5]

This paper, written in 2011, explored the classification and recognition of facial expression through the use of a feedforward backpropagation neural network. This system processed facial data into intelligent distances such as height of eyebrows. The algorithm classified facial expressions into surprise, neutral, sad, disgust, fear, happy and angry. 120 data samples were used for training and 30 samples for testing. The algorithm had a classification accuracy of 96.42%.

### Conclusions from References

When compared to the 2005 report, this current project will use a neural network rather than a fuzzy logic system. This is in order to reduce the reliance on fuzzy rules and human input. The 2011 study also uses neural networks and the high performance achieved is encouraging. The inputs for this current study will be more neutral and less intelligent than either paper. By processing the raw facial landmarks rather than intelligent metrics (such as distance between eyes), the system can also take into account subtle differences in facial features that may not be as obvious to humans. Both reports also implement their own image processing and data collection, but this report focuses on just the neural network classification.

#### Facial Input

Many of the researched papers used intelligent metrics as inputs. For example Kim, Joo, and Park [4] broke the face into three sections, the eye, mouth, and auxiliary section. Each section was analyzed for metrics such as openness of the eye or distance between the nose and mouth. Chaturvedi and Tripathi [6] located 14 facial points and used them to calculate 17 distances. This study will forego processing the points and just enter the full set of points. Using facial landmarks directly is more applicable for a real-time system, since a landmark detection algorithm could feed data directly into the neural network classifier, without any pre-processing. Human interpretation of the data into metrics such as eye openness and distance between eyes may simplify the inputs but may overlook other subtle facial cues of expressions.

#### Computational Intelligence System Base

Various computational intelligence techniques were applied in the researched papers. There were implementations ranging from fuzzy classifiers with Kim, Joo, and Park [4] to Khandait, Thool, and Khandait [5] who used neural networks. In general, all the algorithms were able to achieve accurate results given their defined scope. Since this study seeks to process large data sets for inputs, neural networks would be a better option than fuzzy logic. Khandait, Thool, and Khandait [5] were able to achieve an accuracy of 96.42% using an artificial neural network.

#### Expression selection for output

There were six emotions that were analyzed across the papers which were examined: happiness, surprise, fear, anger, disgust, and sadness. Some studies had less, but all the papers studied happiness, surprise, anger, and sadness. Starostenko et al. [7] included a confusion matrix which demonstrated that surprise and fear were easily confused for each other. In light of these results, this study will analyze the four common emotions of happiness, surprise, anger, and sadness.

# 

# Implementation

The implementation consisted of using the Matlab neural network (nnstart) library for classification, as well as a neural network that was implemented in code ourselves in Matlab. Originally the group planned on using fuzzy logic to specify the fuzziness of facial landmark specifiers such as how high the eyebrows were and how curved the mouth was. However, a decision was made to change to using neural network with the training data. The original plan had also included finding a library for extracting facial landmarks in real time using a web camera. However, it was difficult to find a free or open source library that could provide consistent and accurate results. Instead a database with static data was used.

The database was from a research group from the University of Pittsburgh (Affect Analysis Group). They database called the “Cohn-Kanade AU-Coded Expression Database”, provided images, facial landmarks, and the emotions of 225 images [8]. A script was written in Matlab to display the 68 landmark points (x,y) overtop of the images provided with numbers on each facial landmark.





Figure 1: Landmarks matching for four facial expressions

As can be seen from the images above, the landmark point locations across the images are not all centered about the same position. The x and y positions of the landmarks are normalized and centered around the average of all the points in order to provide a consistent comparison between different faces and emotions.

Instead of using extracted facial measurements like height of eyes, height of mouth, curve of mouth, height of eyebrows, etc as input to the neural network, the group decided to use the landmark points themselves as inputs to the neural network. Therefore, the inputs used were the 136 values making up the 68 landmark points (x1, x2, …, x68, y1, y2, …, y68).

Four output classes were decided upon for emotion classification; happy, surprise, anger and sadness. This is a subset of the number of emotions that current research projects present and was decided up due to having the highest classification accuracy results in papers.

A script was made to iterate through the database of facial landmarks and emotions and generate two matrices for the inputs and targets.

The implementation uses 65% of the exemplars for training and the remaining 35% for testing. The simple nnstart GUI was first used for implementing a classification net with the Matlab library with a single layer with number of neurons. The group’s neural network implementation was also coded in Matlab but only used Matlabs more generic functions such as max(), mean(), size(), length(), zeros(), etc and implemented the neural network using matrix multiplication. Multiple matrices were stored under the same variables for each layer.

tot = total summation of weights{layer} \* output{layer-1} for all nodes entering

node(layer)

output = [Bias; training\_inputs] [Bias; NODES(layer)] [NODES(layer)]

= vector per a layer where

[input\_size(136) + 1(bias)]

[nodes\_hidden\_layer() + 1(bias)] (for each hidden layer) [number\_of\_outputNodes(4) = # of classes]

= activation\_function(tot{layer});

delta = [input\_size(136)] [nodes\_hidden\_layer()] [number\_of\_outputNodes]

= vector per a layer

= if last layer ->

delta{layer} = -1.\*(trainingTargets(:,k)-output{layer}) .\* ((1-output{layer}).\*output{layer});

else

delta{layer} = ((1-output{layer}(2:end)).\*output{layer}(2:end)) .\*

(weights{layer+1}(:,2:end)'\*delta{layer+1});

delta\_weight = [] [neurons x input\_size(137)] [output x (neurons + 1(bias))]

= matrix per a layer where the matrix has the size NODES(layer) by NODES(layer-1) + 1, where the plus one if for the bias.

= delta\_weight{layer} = delta\_weight{layer} + (delta{layer}\*output{layer-1}');

(Offline learning by summing delta\_weigth across exemplars per an epoch

weights = [] [neurons x input\_size(137)] [output x (neurons + 1(bias))]

= matrix per a layer where the matrix has the size NODES(layer) by NODES(layer-1) + 1, where the plus one if for the bias.

= weights{layer} = weights{layer} + (-1.\*eta.\*delta\_weight{layer});

(Offline learning by summing weights across exemplars per an epoch

The code version was implemented using offline (batch) learning to improve performance as weights were updated per epoch and not for each exemplar. As well, it was found that without offline learning, the error (cumulative of outputs minus targets) would never decrease and would converge to its initial value (large error). As well, the mean value across each input point (mean(x1), mean(x2), etc) and the mean across the points in an image were subtracted from the inputs to center the points at the zero position on the axis of a 136 axis graph to prevent the gradient descent from moving in the incorrect direction. Without subtracting the mean values, the error would not decrease, similarly to the results of online learning.

The group also implemented momentum to speed up the convergence of the gradient descent. However, in certain cases this resulted in worse accuracy result. This was likely due to the values chosen for the learning rate, and momentum coefficient. While the system was able to descend quicker along the gradient, and was faster to converge, it is still important to choose appropriate parameter values for the neural network. For the purposes of this study, systems were trained with and without momentum, and the best results were used.

# Results

To achieve the best results, a number of different neural network configurations were investigated. The number of neurons, number of hidden layers and input size (landmarks used) were varied to find the configuration that would provide the highest classification accuracy. For each test, the system was trained to a maximum of 5000 epochs, or until the total cumulative error for the epoch went below a value of 0.01. After selecting the best solution, the real time performance of the classifier is tested for its viability as a real-time emotion detection system.

## Varying Number of Neurons

The number of neurons in each hidden layer can greatly affect the classification accuracy of the neural network. A single hidden layer network was created with both the Matlab nnstart tool as well as the group’s custom written neural network. For both systems, the number of neurons was varied between 1 and 100. The classification accuracy results can be seen in Figures 3 and 4 below.

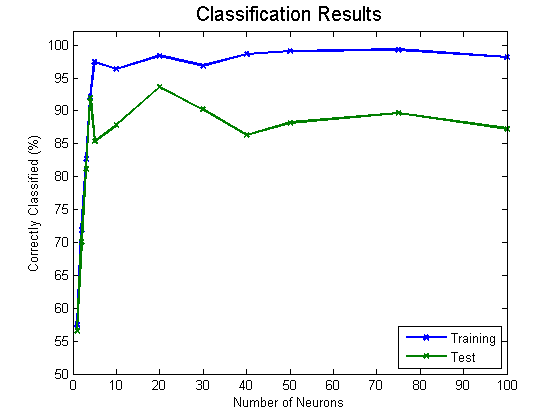


Figure 3: Matlab nnstart, Varying Number of Neurons Results

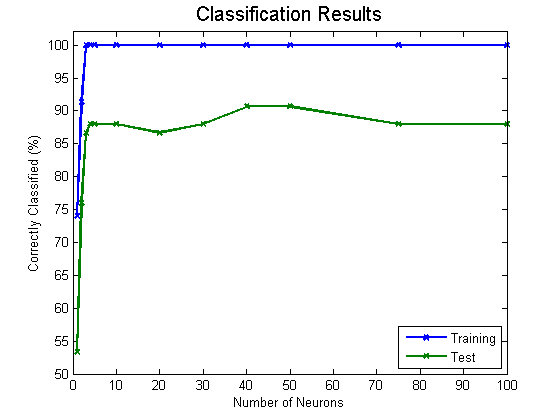


Figure 4: Group’s Custom Neural Network, Varying Number of Neurons Results

For the Matlab neural network toolbox, a classifier with 20 neurons in the hidden layer provided the best classification accuracy of approximately 93% for the test samples. For the group’s custom network, using 50 neurons in the hidden layer provided the best classification accuracy of approximately 91% for the test samples.

## Varying Number of Hidden Layers

The number of hidden layers within the neural network can also affect the classification accuracy of the network. The effect of varying the number of hidden layers was investigated for systems with 5, 20, and 50 neurons per hidden layer. The table below summarizes the results that were obtained.

Matlab Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Neurons Per Layer | 5 Neurons |  | 20 Neurons |  | 50 Neurons |  |
| Number of Hidden Layers | Training | Test | Training | Test | Training | Test |
| 1 | **89 %** | **81 %** | **96 %** | **92 %** | **93 %** | **81 %** |
| 2 | 68 **%** | 65 **%** | 100 **%** | 86 **%** | 89 **%** | 80 **%** |
| 3 | 39 **%** | 33 **%** | 93 **%** | 90 **%** | 80 **%** | 77 **%** |

Group’s Custom Network Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Neurons Per Layer | 5 Neurons |  | 20 Neurons |  | 50 Neurons |  |
| Number of Hidden Layers | Training | Test | Training | Test | Training | Test |
| 1 | **100 %** | **88 %** | 100 % | 87 % | **100 %** | **91 %** |
| 2 | 91 % | 76 % | **100 %** | **89 %** | 100 % | 88 % |
| 3 | 91 % | 75 % | 100 % | 83 % | 100 % | 87 % |

Figure 5: Accuracy results for varying number of nodes and number of hidden layers

The results of varying the number of layers for Matlab showed that increasing the number of hidden layers did not significantly improve the performance of the neural network, and was actually detrimental in certain systems. The Matlab neural network that provided the best result was the single hidden layer, 20 neuron system. In the group’s custom neural network, increasing the number of hidden layers also did not show a clear improvement. The best system was a single layer, 50 neuron classification neural network.

## Cumulative Error and Classification Results

After finding the neural network that provided the best accuracy, the cumulative error and classification results was plotted against the number of training epochs for the group’s custom neural network. For the system with the best classification accuracy, the error and classification results can be seen in Figure 6 below.

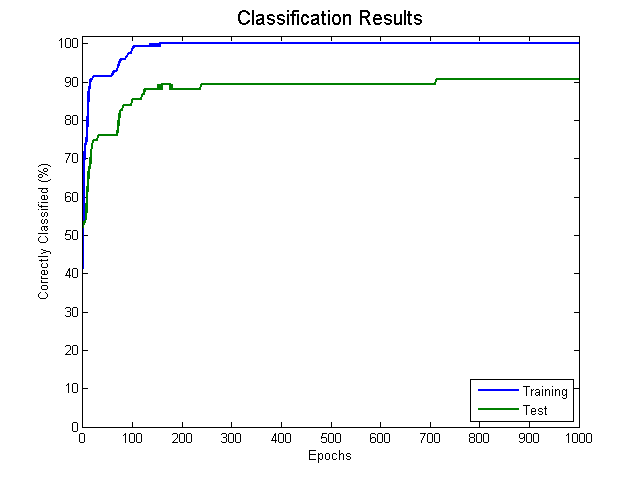
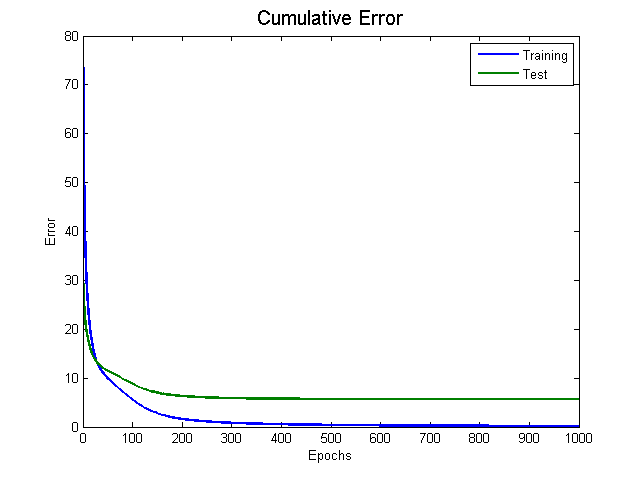
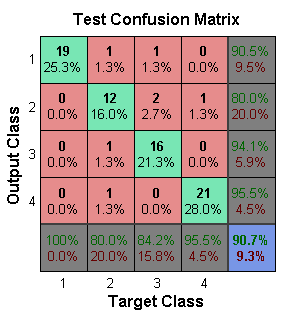
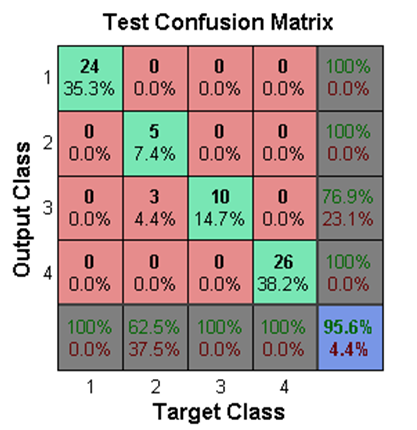


Figure 6: Training versus Testing results for error and classification

As shown in the plots above, the cumulative error for both the training and test samples decreases over each training epoch. The classification accuracy can also been seen to improve as the training error and test error decrease. The system was trained for 5000 epochs, however, no significant improvements were noted after the first 1000 epochs.

The best results from the Matlab neural network toolbox were then compared with the best results from the group’s custom network. Both systems used a single hidden layer, with 20 neurons for the Matlab version and 50 neurons for the group’s version. The confusion matrices for both results can be seen below.



|  |  |
| --- | --- |
| Figure 7: Matlab - 20 Neurons Best Result | Figure 8: Custom Network - 50 Neurons Best Result |

The 20 neuron Matlab classifier achieved a higher overall test accuracy of 95.6%. However, it had less success distinguishing between target classes 2 and 3 (Sadness and Anger), while the custom network had over 80% accuracy for each emotion, and an overall accuracy of 90.7%

## Varying Input Size

The group was interested in the effects of varying the input size, or in other words the effects of the facial landmarks used on the accuracy of emotion classification. The reason for this is was to minimize the input size required, which reduces memory size and processing time for a facial landmark recognition system, and provides a better understanding of which parts of the face provide the most emotional information. The results are presented in the tables below.

Custom Implementation Version

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **All Points** | **!(Jaw)** | **!(Eyebrows)** | **!(Eyes)** | **!(Nose)** | **!(Mouth)** |
| Training | 100 | 100 | 100 | 100 | 100 | 97.9 |
| Testing | 88.6 | 91.1 | 89.9 | 89.9 | 86.1 | 70.9 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **!(Jaw, Eyebrows)** | **!(Jaw, Eyebrows, Eyes)** | **!(Jaw, Eyebrows, Eyes, Nose)** |
| Training | 100 | 100 | 100 |
| Testing | 91.1 | 92.4 | 93.7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Only Jaw** | **Only Mouth** | **Only Eyebrows** | **Only Eyes** | **Only Nose** |
| Training | 96.6 | 100 | 97.3 | 87.7 | 76.7 |
| Testing | 50.6 | 93.7 | 43 | 59.5 | 43 |

MATLAB Library Version

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **All Points** | | **!(Jaw)** | | **!(Eyebrows)** | | **!(Eyes)** | | **!(Nose)** | | **!(Mouth)** | |
| Neurons | 20 | 50 | 20 | 50 | 20 | 50 | 20 | 50 | 20 | 50 | 20 | 50 |
| Training | 98.6 | 98.6 | 97.3 | 97.9 | 98.6 | 93.8 | 100 | 95.2 | 97.3 | 100 | 85.6 | 93.8 |
| Testing | 91.2 | 83.8 | 97.1 | 95.6 | 83.8 | 86.8 | 94.1 | 88.2 | 88.2 | 95.6 | 66.2 | 75 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **!(Jaw, Eyebrows)** | | **!(Jaw, Eyebrows, Eyes)** | | **!(Jaw, Eyebrows, Eyes, Nose)** | |
| Neurons | 20 | 50 | 20 | 50 | 20 | 50 |
| Training | 97.3 | 93.2 | 94.5 | 93.8 | 91.8 | 93.2 |
| Testing | 92.6 | 83.8 | 88.2 | 80.9 | 86.8 | 92.6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Only Jaw** | | **Only Mouth** | | **Only Eyebrows** | | **Only Eyes** | | **Only Nose** | |
| Neurons | 20 | 50 | 20 | 50 | 20 | 50 | 20 | 50 | 20 | 50 |
| Training | 39 | 65.1 | 91.8 | 93.2 | 64.4 | 60.3 | 34.2 | 71.2 | 53.4 | 33.6 |
| Testing | 38.2 | 54.4 | 86.8 | 92.6 | 58.8 | 66.2 | 32.4 | 50.0 | 41.2 | 30.9 |

Figure 9: Accuracy results with removed features for both implementations.

## The results from our code implementation appear similar to the Matlab results. The results demonstrate that the jaw is the least import attribute as removing those landmarks did not significantly decrease the classification accuracy, which was 91%. The mouth can be seen to be the most important as the accuracy with the mouth landmarks removed is the worst at 70.9%. However, for real time applications where the mouth would be moving, removing the mouth still provides a fairly reasonable estimate. The best result for the code implementation was using only the mouth at 93.7%. The best result using the Matlab library was with the removal of the jaw at around 97% for 20 neurons or 95.6% for 50 neurons.

It is hard to evaluate the order of importance for the facial features in emotion detection, though it can be said that the mouth is the most important but not entirely necessary. This may be due to the number of facial points that the mouth uses, 40 points, the most from all the features, as seen in the table below.

Figure 10: Number of landmarks corresponding to each facial feature

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Jaw** | **Mouth** | **Eyebrows** | **Eyes** | **Nose** |
| Number of Data Points in category | (1:17)\*2 =  34 | (49:68)\*2 = 40 | (18:27)\*2 = 20 | (37:48)\*2 = 24 | (28:36)\*2 = 18 |

## Real Time Performance and Memory Size

Real time refers to details pertaining to the requirements if the system for detecting facial emotion was implemented on a system (ex. microprocessor/hardware/fpga, mobile phone, website, robot, etc with use of a camera) and performed emotion detection on the current user on the spot. However, for real time implementation, software will be required to extract the facial landmarks on the current image or video captured by the camera.

Considering the size of memory required for a real time implementation, disregarding the facial landmark extraction part of the code, leaves the size of the input, size of the feedforward neural network code, and size of the matrix containing the neural network weights (for a single hidden layer at 50 neurons).

* Size of input (for all landmarks) = 2 KB
* Size of feedForward Code = 3 KB
* SIze of Neural Network Weights = 54 KB

As seen above, the total memory size required is approximately 60 KB, which is very small and could be stored on many types of devices. Hence, memory should not be a problem in implementing a real time version of the facial emotion detection system.

In order to perform real time detection, the system would theoretically extract the facial landmarks from a video input, and pass the landmark locations to the classifier. The classification process of the network should therefore be as quick as possible, in order to allow ample time for facial landmark extraction. For both the Matlab and the group’s solution, the time to classify a single input was determined. The time for classifying the input without the mouth landmarks was also tested. Each test was run over the full set of 225 samples, and the times were averaged and are shown in the table below.

|  |  |  |
| --- | --- | --- |
|  | Single Input | Mouth Removed |
| MATLAB | 6.3 ms | 6.1 ms |
| Custom | **0.038 ms** | **0.036 ms** |

Figure 11: Classification Performance comparison for MATLAB and custom implementation

As shown in the table above, the Matlab neural network takes over 6 ms to classify a single input, while the group’s solution is able to classify a single input in approximately 0.04 ms. The Matlab implementation of the neural network was further investigated and found to contain significant overhead for classifying a single input, as it is able to classify an array of inputs in approximately the same time as a single one. However, this is not desirable for a real-time system. The group’s solution performed much quicker since the weights of the neural network are represented as a matrix of values, and the calculations can be done very quickly. Removing the mouth from the inputs did not reduce the classification time by a significant amount in either case. While both solutions are feasible for a real time solution, the group’s solution was able to classify an input much quicker, allowing for more time to extract facial landmarks on each video frame.

# Conclusion

In conclusion, both the Matlab library and neural network code created by the group provide greater than 90% accuracy in classifying between 4 emotions. The Matlab library was found to be best for 20 neurons in single layer, and the group’s version was best at 50 neurons. As well, increasing the number layers was found to not help to improve accuracy, the accuracy was actually found to get worse.

Varying the input size to the neural network and identifying the effects specific facial landmarks have on accuracy were found through experimentation. The result demonstrated that the jaw is the least important attribute since without it the accuracy was still 91%. The mouth was found to be the most important in discerning emotions, as with only the mouth the accuracy was at 93.7% in our coded version and 86.8% in the Matlab library version. Removing the mouth information was still found to leave reasonably good accuracy at 70%, hence it is feasible to have the neural network run in real time without the mouth to detect the emotion of people while talking.

The best result for the group’s coded version of the neural network was when using only the mouth, however the best result for the Matlab library was with the removal of the jaw which resulted in 97% accuracy. The reason the mouth was found to be so important might have been due to the mouth taking the majority of input points, 29% of the input points (40 out of 136). This result is better than the fuzzy logic implementation of Kim, Joo, and Park [4] and closely matches the results of Khandait, Thool, and Khandait [5] of 96.42%. It should be noted that this study did simplify the problem by having a minimal amount of facial expression outputs so a direct comparison of accuracy is not truly valid.

The overall memory requirements for this system were found to be small, only requiring approximately 60 KB of space. Real time performance testing showed that the group’s neural network was able to classify a single input much quicker than the Matlab implementation. Classification of a single input took approximately 0.04 ms, which is definitely fast enough for implementation in a real time system. The next step in this project would be to integrate a real time facial landmark detection system in order to provide classification results in real time.

# 

# 

# References

|  |  |
| --- | --- |
| [1] | S. W. Littlejohn and K. A. Foss, Encyclopedia of Communication Theory, SAGE Publications, Inc., 2009 . |
| [2] | Apple Inc., "Siri," Apple Inc., 2015. [Online]. Available: http://www.apple.com/ios/siri/. [Accessed 2 April 2015]. |
| [3] | H. Hodson, "Speech analyser monitors emotion for call centres," *New Scientist,* p. 1, 8 May 2014. |
| [4] | M. H. Kim, Y. H. Joo and J. B. Park, "Emotion Detection Algorithm Using Frontal Face Image," International Conference on Computer Application in Shipbuilding, Gyeonggi-Do, 2005. |
| [5] | S. Khandait, R. Thool and P. Khandait, "Automatic Facial Feature Extraction and Expression," *International Journal of Advanced Computer Science and Applications,* vol. 2, no. 1, p. 6, 2011. |
| [6] | A. Chaturvedi and A. Tripathi, "Emotion Recognition using Fuzzy Rule-based System," *International Journal of Computer Applications,* vol. 93, no. 11, p. 4, 2014. |
| [7] | O. Starostenko, R. Contreras and V. A. Aquino, "A Fuzzy Reasoning Model for Recognition of Facial Expressions," *Computación y Sistemas,* vol. 15, no. 2, p. 18, 2011. |
| [8] | J. F. Cohn and T. Kanade, "Cohn-Kanade AU-Coded Expression Database," University of Pittsburgh, Pittsburgh, 2015. |