Sign-Language Alphabet Gesture Recognition

Timothy Cassidy

B.Sc.(Hons) in Software Development

April 26, 2019

Final Year Project

Advised by: Dr. Patrick Mannion

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract There is a large communication gap between the Deaf/non-verbal community and the hearing/verbal majority. Innovations in computer vision and gesture recognition, with the aid of Neural Networks, are being used to solve this communication issue. The aim of this project was to develop an application that could determine the potential use of an Artificial Neural Network and computer vision as a medium to bridge the communication restrictions between non-verbal sign-language users and persons with no knowledge of sign-language communication. The goal of this project was to develop an application that could recognize sign-language gestures, using the users webcam to capture visual input. Initially, the visual input is processed using a Haar Cascade Classifier, this was used to excerpt a skin tone range. A static region of interest was defined in order to narrow the area of prediction. The visual input, captured from the region of interest, was prepared by extracting any features that were within the defined skin tone range. The gesture recognition was preformed by a hosted, Server, Artificial Neural Network which processed the prepared visual input and made a prediction based on this input, the prediction made was based on the American Sign-Language alphabet. The result of the prediction is then passed to the client, which displays the predicted result. This application demonstrates the applicability of an Artificial Neural Network to solve the identified communication problem.

Acknowledgements I would like to acknowledge and thank my supervisor Dr. Patrick Mannion for the time and effort he put into helping me throughout this project. I would like to also thank all our lectures at Galway-Mayo Institute of Technology for equipping me with the skills and knowledge in the past 4 years that has enabled me to complete this project

Author Timothy Cassidy - 4^{th} Year Software Development Student

Chapter 1

Introduction

1.0.1 Project Aim

The aim of this project was to develop an application that could recognize sign-language gestures, over the twenty-six letters of the English alphabet, this was achieved using a Neural Network. Sign-language is a language set that employs visual signs to communicate, these visual signs, or gestures, are made with the hands and other movements, these can include facial expressions, the posture of the body and the body language used when signing. Sign-language is primarily used by people who are deaf or non-verbal.

1.0.2 Sign Language Families

Sign-languages, like spoken languages, belong to language families, the signed languages often share little to no relation to the spoken language of their name. A selection of signed language families include BANZSL, which is the British SL(sign-language), Australian SL and the New Zeland SL, another is the French SL, this includes the French SL, Danish SL, Italian SL, American SL and the Irish SL. Because ASL and ISL are related languages, their manual alphabets are quite similar, but only some of their signs are similar.

1.0.3 American Sign Language

There are many different forms of sign language, as mentioned above, for the purpose of this project the focus will be on the American Sign-language, or ASL. When using ASL, sentences tend to follow Subject-Verb-Object or Subject-Verb structure, even though this may overlap, in some cases, with the English language word order this word ordering is distinct from English word order. ASL does not use words to describe or indicate the state of

being, for example: am, is, are, was, were. Additionally, ASL does not use articles, for example: a, an, the.

1.0.4 Dataset

The American Sign-language variant, ASL, was chosen because of the availability of datasets, which will be used to train the Neural Network. The dataset chosen needed to contain a large quantity of images of various angles per gesture. A Neural Network is a set of algorithms that are designed for pattern recognition, these algorithms are usually modeled based on a loose interpretation of the human brain. Neural Networks can me tailored to receive any kind of numerical data contained in a vector, this data can be any type of real-world data, such as sound, images or text. Any input into a Neural Network must be translated into a numerical vector. A Neural Network is trained on a dataset, which is a collection of correctly labeled, sorted data. The dataset used can have a major impact on both the training accuracy and the prediction output.

1.0.5 Neural Network

Neural networks are primarily used to classify. A Neural Network, simply, groups unlabeled input data according to the similarities among the dataset that were discovered during training. The machine learning method used is known as supervised learning, this method will classify data only when it has been trained on a labeled dataset, the other method of learning is unsupervised learning, which is used to discover possible patterns in input data without and dataset training.

1.0.6 Colour Model

The input data for generating predictions is captured via the users webcam. A region of interest has been defined where hand gestures will be captured, this is represented by a boxed region overlay on the webcam output area. The data captured is reduced to HSV values, HSV meaning Hue Saturation Value, the HSV colour model is commonly used in computer vision as it can be used to separate image luminance from colour information. The HSV colour model was used instead of the RGB colour model, RGB meaning Red Green Blue, because the RGB colour model cannot separate image luminance from colour information because the values are co-related.

1.0.7 Research Areas

Computer vision, machine learning and gesture recognition are all quite interesting topics, the applications of these fields of research are extremely broad and cover a lot of aspects of daily life, from face detection in our smart-phones to gesture control in our televisions. This project attempts to showcases the potential use of these technologies to bridge the communication gap between non-verbal sign-language users and people that have no sign-language education.

1.0.8 Objectives

The objectives for this project were as follows:

- To train an artificial neural network to recognize the American Sign Language alphabet.
- To create a real-time gesture recognition application interface.
- To expand the artificial neural network beyond the ASL alphabet.
- To host the artificial neural network to allow for threaded multiple-user access.

1.0.9 Summary Of Chapters

Below is a brief description of the chapters to follow:

The methodology section will cover the approach taken, the planning of the project, supervisor meetings and the testing performed.

The technology review section will cover the conceptual view of the technologies used and all research undertaken.

The system design section will cover the application of the research from the technology review section and it will provide a detailed description of the system architecture and how the system is coupled.

The system evaluation section will contain the evaluation of the project in contrast to the proposed objectives, the robustness of the project will also be analyzed along with the limitations and advantages of the technologies used.

The conclusion section will discuss the findings of the system evaluation and additionally will propose plans for future development.

1.0.10 GitHub Repository URL

https://github.com/Trojan-Hawk/Sign-Language-Recognition

Chapter 2

Methodology

2.1 Approach

This section will outline the approach that was taken in order to develop this project. The topic that will be covered is the methodology utilized and how it was implemented into the research and development of the project. The purpose of this section is to inform the person reading this of the development process and how the software was delivered at it's final stage.

2.1.1 Project Plan

As the project was started with an open development subject, the initial stages of the project consisted of researching possible topics of development. When the subject area was decided upon the development stage began, initially the approach taken here was Ad Hoc, when necessary or needed. This felt quite unstructured and a rough development plan was required in order to ensure we stayed on schedule. The project plan, which was flexible due to accommodate possible development changes, was initially planned using an Agile model. The Agile model is a programmer's way to planning and project management, Agile is a project management process where requirements and output evolve through the collaborative effort of the software development team.

2.1.2 Communication

Meetings with our supervisor were scheduled for Wednesday evening every week, this enabled us to stay on track and to have a short informal presentation of progress so far and to discuss the requirements for the next meeting. Outside of project meetings we communicated using the Facebook

Messenger application initially and we used GitHub for versioning control of the development of the project. Due to Facebook Messenger's file sharing size restrictions and the constant distraction of a social media platform, we decided it was best to communicate on a platform free of distractions and dedicated to project discussion, for this we used Slack. To combat the file sharing restriction we used Google Drive as the file sharing platform.

2.1.3 Plan Revision

Due to unforeseen circumstances my project partner had to defer until the next academic year which left the development and additional research of the project solely in my hands. This change in the project team meant there had to be a change in the project scope and development method. After consulting with my project supervisor, the project scope was reduced from sign-language gesture recognition on both the alphabet and a selection of common words to sign-language gesture recognition over the alphabet. The project development model also changed from an Agile method to one that closely resembled the waterfall model, the waterfall model is a linear, sequential approach to development, this was due to only one person developing the project.

Below is a brief description of the communication technologies used:

2.1.4 GitHub

GitHub was used to manage the Project and to allow version control. There are a multitude of different project management and version control tools available on the software market, each with their own specific functions and tools. GitHub was chosen due to our familiarity with it as we have used it for multiple past projects and because it offers a simple, user-friendly and effective approach to project management and version control.

2.1.5 Slack

Slack was used as an alternative to the Facebook Messenger application, and was useful due to the lack of distractions that came with using the Facebook Messenger social media platform. Slack is a cloud-based set of proprietary team collaboration tools and services, although there are an astonishing number of these tools and services we only focused on the communication aspect of this application.

2.1.6 Google Drive

Google Drive was used mainly used to share any relevant research papers we found and to share any necessary files that we did not want cluttering our GitHub. Google Drive offered a storage space of 15GB, which was more than enough for what was required. The best feature of Google Drive and most used by us was the shared document editing, this allowed us to leave notes on each research paper highlighting important elements.

2.1.7 Facebook Messenger

Facebook Messenger was used initially as a communication platform for the project but this proved to be very distracting, as it is a social media platform. The main reason we decided to change communication platform was due to the file sharing restriction of 25MB.

2.2 Monthly Work Breakdown

2.2.1 October

During the first two weeks we mainly focused on research, all time was dedicated to narrowing down the subject area of the project and compiling relevant research material. While deciding on a subject to base our project we discussed technologies we had already used and their potential application. We both had an interest in the Python programming language, which we only had a small base of knowledge, so we decided to use Python in order to improve our dept of knowledge of the language. We also discussed the possibility of incorporating some machine learning aspects into the project as we both had an interest in this field, we decided to structure the project around computer vision. After we decided on the language to use and we identified that we would like to center the project around computer vision we decide to brainstorm potential project ideas, the result of which was a sign-language gesture recognition application.

2.2.2 November

In order to develop a gesture recognition application we needed to implement an artificial neural network, this would need a suitable dataset in order to be trained properly. From this we identified the first step was to acquire a suitable dataset of optimal quality. In order to choose a dataset we first had to decide what variant of sign-language to use. After researching the different

variants of sign-language we decided to use the ISL, Irish sign-language, variant but it proved too difficult to find a dataset of a good quality. We ended up choosing the ASL, American sign-language, variant because of the availability of high quality datasets.

2.2.3 December

The initial work began on developing a convolutional neural network using the dataset sourced in the previous month. The dataset was divided into two separate parts, eighty-three percent of the dataset for training and the remaining seventeen percent for testing, these are then further divided into actual image data and the labels associated with the images. This rough draft of the Convolutional Neural Network, or CNN, was quite in-accurate, when tested with the testing portion of the dataset the error rate, percent of misidentified images, was seventy-nine percent. The next task was to bring this error rate down to an acceptable level.

2.2.4 January

This is the stage of the project where the team project became a solo venture. The main focus of this month was on the revision of the project objectives, which had to be redefined to deal with the change. As the GitHub repository we had initially been using was on my project partners account, of which I was a collaborator, a new GitHub repository was created. At this stage the error rate of the CNN was still very high. The process of attempting to reducing this error rate involved, performing minor changes, re-training the CNN and recording the error rate.

2.2.5 February

At this stage of the project the CNN error rate was greater than sixty percent. Due to the completely in-accurate non-improving results, the artificial neural network needed to be re-designed. The result of the re-design stage was an artificial neural network with a significant reduction in complexity. The reduction in complexity proved fortuitous as the initial error rate was below twenty percent. Research began on the graphical user interface, or GUI, element of the project. As the project was executed via the terminal up until now, this interface element of the project needed development.

2.2.6 March

Initially my supervisor pointed me in the direction of Ionic 3 to develop the GUI aspect of the project. As I had previously worked with both Ionic 1 and Ionic 2 this seemed like a good choice, however when I had a working prototype it raised some issues, for more information on this issue see section 4.2.1. This then led me to use tkinter, which is a GUI package for Python. Towards the second half of the month an Amazon Web Services, or AWS, server was created, initially the method I was attempting proved to be ineffective and a new method of serving the application was adopted, for more information on this issue see section 4.2.2.

2.2.7 April

During the first week of this month a server issue lead to the shutting down of the server instance and the creation of a new instance, for more information on this issue see section 4.2.3. The GUI element of the client application was completed.

2.3 Testing

Testing, with regards to the neural network, was preformed mainly by referring to the error rate of the neural network after training had been completed. This was done, after training, by using the **evaluate** method from the Keras library, this method evaluates the trained neural network based on the testing dataset. The will be the first time the neural network has observed the testing dataset, this is to ensure the data is new and to get an accurate error report. The final error rate of the neural network was 0.31%, which means it misidentified less that half a percent of the test dataset data.

Testing of the GUI and all other elements was done Ad Hoc during development, this was mainly due to their being only one member developing and testing. Automated testing was not applicable to any of the developed features of the application so resources were not invested into automated testing of any type.

Chapter 3

Technology Review

3.1 Technologies Used

This chapter will discuss, in detail, the different technologies incorporated into this project. The chapter will additionally cover the rational and the research behind each choice.

3.1.1 Cloud

For the cloud server element of this project there were two obvious choices, these were Amazon Web Services, AWS, or the Google Cloud Platform, GCP. These two providers are at the forefront of supplying cloud services. AWS is the clear leader in terms of the numbers of customers and products, this is mainly due to the five years running start ahead of all other cloud services. While GCP have a number of services available and are serious competitor of AWS, several GCP features are still in alpha or beta stage and this could cause issues for long running projects if the API or behaviour change. AWS do have an initial information overload associated with their service because of the broad range of products they offer. This can be quite overwhelming to new users. Due to their clear lead and dominance over the market and because I was quite familiar with both technologies I decide to choose AWS to deploy my server, ultimately the decision was based on preference.

3.1.2 Programming Tools

The programming language used was Python, this is mainly due to the popularity of this language in the field of machine learning and as a result of this there are a wide range of projects available for research. Another reason this language was chosen was due to the previous experience I had and because

it is a very easy language to learn due to it's intuitive syntax.

Notepad++

Notepad++ is text and source code editor developed for use on a Microsoft Windows system. Initially I had used Notepad++ as a code editor and then using the standard command terminal on my Windows system to execute developed programs, this however proved to be quite time consuming. To reduce the time spent error checking and executing in the above method it was decided to use an IDE, integrated development environment.

Integrated Development Environment

The choice of IDE was reduced to the following three applications:
The first potential choice was Visual Studio, which is an IDE from Microsoft, is used to develop many programs for Microsoft Windows, as well as web sites, web applications and web services. The drawbacks of using Visual Studio are it is a very cumbersome application and my experience with this IDE is it can be quite slow.
The next possibility was to use Eclipse, which is the most widely used IDE for Java development, it contains an extensible plug-in system for customizing the development environment. This plug-in system can be extended to allow for the development of Python. The downside of using Eclipse is if you are not familiar with there can be a steep initial learning curve.
The final choice of IDE was PyCharm, this is a Python specific IDE which can edit, run and debug Python code. The only drawback with using PyCharm is setting up the Python environment, which can be tedious.
The chosen IDE was PyCharm, this was mainly due to it's singular focus on developing Python code and the availability of an out-of-the-box debugger.

3.1.3 Python Libraries Used

This section covers the Python packages used, the rational behind their use and which features were used. This section is divide up into client side and server side packages used, although there are some overlap in package use they will not be repeated.

Client Side

Below are the packages used on the client side application of this project:

OpenCV is a library of packages mainly focused on real-time computer vision. OpenCV was initially developed by Intel. The OpenCV library is cross-platform and completely open source. The elements of this library used were font-style, for displaying text on screen, video-capture, which got a handle on the current frame from the webcam, flip, which was used to flip an image frame on the Y-axis, converting colour, this was used to convert the colour of the image array to and from RGB, BGR(reverse of RGB) and HSV, rectangle, which was used to draw a rectangle over the frame displayed to the user to signify the area where gestures are captured, inRange, which was used to extract the elements in the image array within the specified range, dilate, which was used to used to accentuate features of the frame supplied and GaussianBlur, which applied a specified blur to the frame which would allow a more generalized feature image to be passed to the artificial neural network.

Numpy is a library which provides support for large, multi-dimensional arrays and matrices, as well as a large collection of mathematical functions to operate on arrays. The elements used of this library were unsigned integer conversion, array conversion and ones, which was used to fill an array with the integer value of one.

Tkinter a library is a Python binding to the Tk GUI toolkit, it is Python's primary GUI development library. The elements of this library used canvas, which defined a canvas for GUI elements to be drawn on, frame, not to be confused with OpenCV image frame this was used to place image elements on the canvas, slider controls, these were used to manually define the HSV colour range to be extracted, and button elements, these were used to allow the user to automatically re-define the HSV range based on the result from the facial recognition element.

Socket is a library which allows Socket programming, this is a way of connecting two nodes on a network. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other on the particular port to form a connection, this then allows the transfer of data to and from both nodes. Blosc is high performance compressor, according to them "It has been designed to transmit data to the processor cache faster than the traditional, non-compressed, direct memory fetch approach"[1]. This compression method was chosen due to its high speed compression rate which

facilitated faster data transfer between client and server.

Pickle is a Python library used for loss-less serializing and de-serializing Python objects. Pickled objects can be saved on disk, which was used to load in the training and testing dataset data and in order to rectify the normalization issue, see section 4.2.4. Simply, pickling is a way of converting a Python object into a character stream.

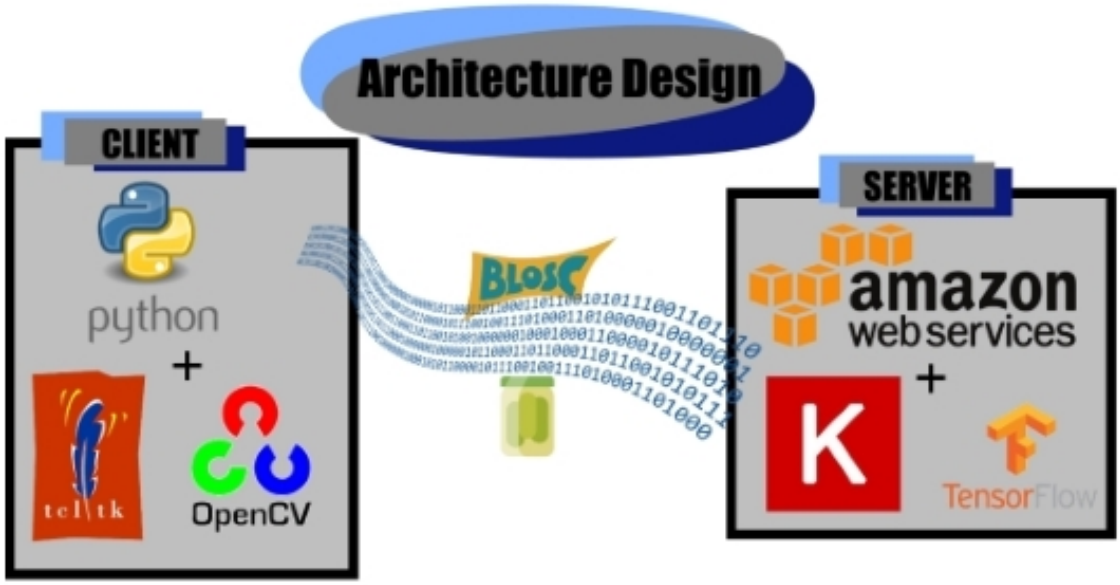
Server Side

OS library provides access to the operating systems dependent functionality. The main functions of this package used were to provide access to the file system in order to access the files and to save files, the path sub-package was also used to verify the existence of specific files.

Keras is an open-source neural-network library written in Python. It is able to run on top of TensorFlow back end. TensorFlow is a library for quick numerical computing, it was developed and released by Google. The Keras library provides an abstraction from the intricacies of the TensorFlow package, this allows a developer that is inexperienced in machine learning to develop an artificial neural network and gain an understanding for the core concepts.

Chapter 4

System Design

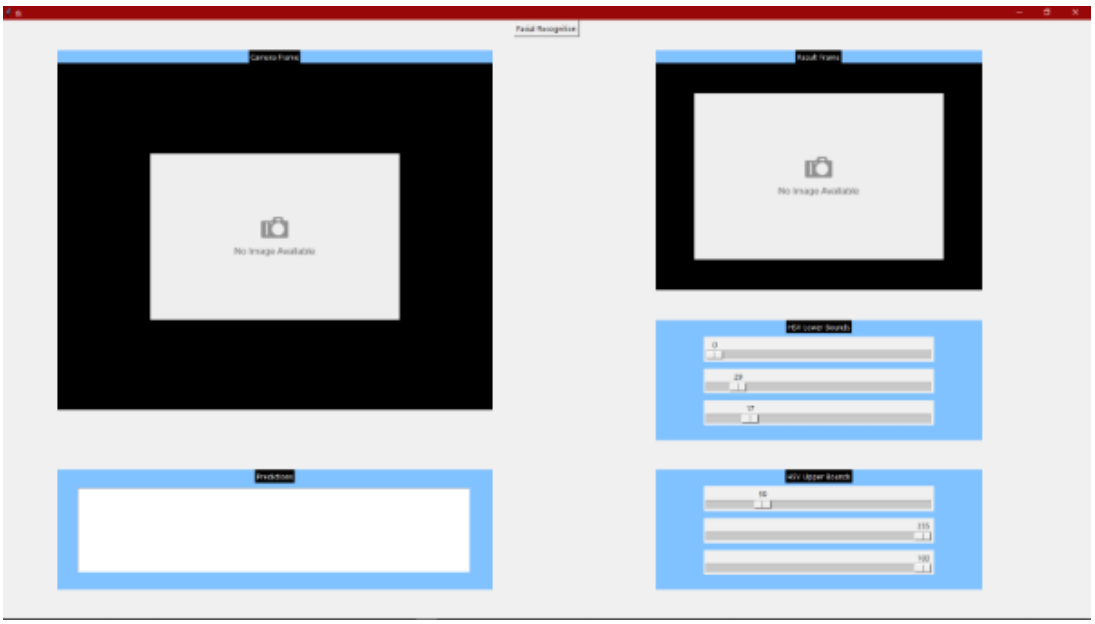


The architecture of this project is a client server relationship. The client utilizes an array technologies to facilitate the capture and presentation of web-cam data. The client also incorporates a graphical interface at the presentation layer. The web-cam data is received per frame. The captured web-cam frame is processed and a region of interest is extracted and delivered to the server via socket communication. Before the data is sent to the server it is compressed using the Blosc compression algorithm and then it is serialized using the Pickle library. When the server has received the data it is un-pickled, de-serialized, and then de-compressed. The frame data is then sent to the artificial neural network to be processed, the result of this is the

encoded, using the Sklearn library, in order to convert it into a single ASCII letter representation.

4.1 System Structure

4.1.1 Graphical User-Interface

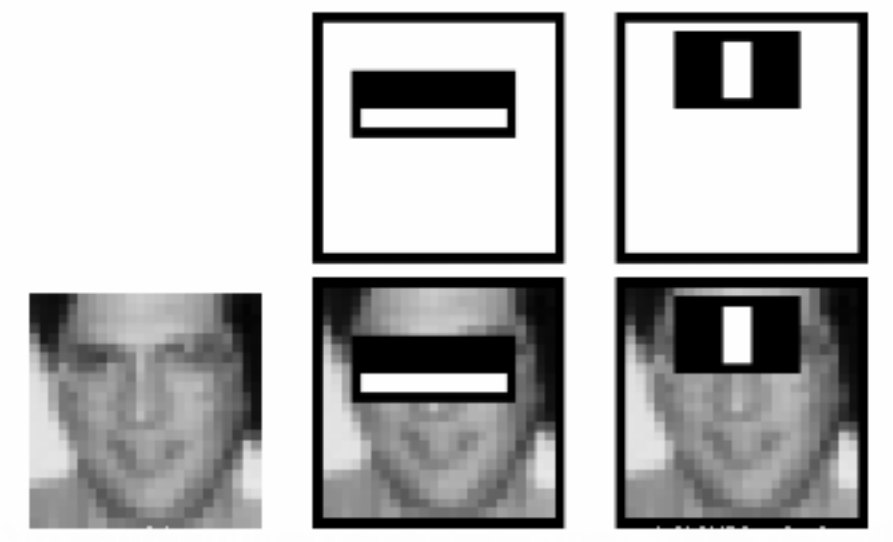


The image above shows the project graphical user interface, this was achieved using the Python package Tkinter. The first element, located in the top-left, is the web-cam output frame which is where the web-cam data is displayed. The element directly underneath this is the output frame, this is frame is where the ASCII output is displayed to the user. The element located in the top-right of the image is the result frame, this is responsible for displaying the extracted region of interest from the web-cam data. Below the result frame are two additional frames, these are the lower and upper bound HSV sliders. These sliders are used to control the upper and lower HSV values that will be extracted from the region of interest. The final element is the button located at the center-top, this button when pressed re-defines the HSV range using the haar cascade facial recognition method(see section 4.1.3).

4.1.2 Web Camera Capture

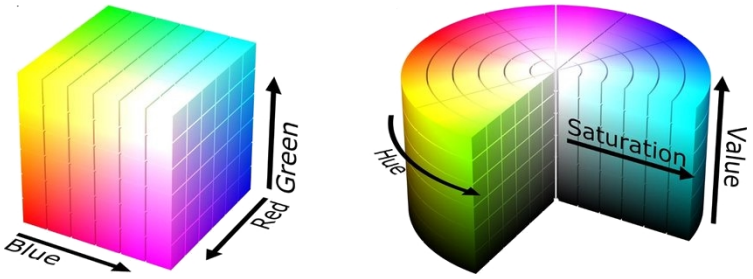
The web-cam capture was preformed using the OpenCV **VideoCapture** method. This method can be given a video file or an integer specifying which web-cam to use, if your device has only one web-cam a zero value can be passed to access the first camera available. The VideoCapture object can then be polled to capture data frame-by-frame by using the **read** method. Before exiting the application the VideoCapture objects handle on the web-cam stream is released in order to avoid a resource-leak.

4.1.3 Facial Recognition



The method of facial recognition used was the Haar Cascade algorithm. This method performs object detection using Haar feature-based cascade classifiers, the features are single values extracted by subtracting the sum of pixels contained within the white segment(shown above) from the sum of the pixels contained within the black segment. As can be seen from the above example, most of the image features are irrelevant. These irrelevant features are removed by using the concept of cascade classifiers, rather than applying all 6000 face detection features on an image features are organized into different stages of classification and are applied in succession. The first few stages contain a much smaller range of features required before advancing to the next stage of classification, this reduces the computational overhead associated by removing irrelevant features early and only applies the more exhaustive classification features to relevant segments.

4.1.4 HSV Value Selection



The data frames captured from the web-cam are presented using the RGB(red green blue) colour model, this presents a problem due to the limitation of this colour model. As can be deduced from the graphic above, the RGB colour model is lacking the colour luminance range, this is because the RGB model's values are co-related and thus cannot be used to separate colour information from colour luminance. The reason behind choosing the HSV(hue saturation value) colour model was to overcome the limitation of the RGB colour model as the HSV colour model has the ability to separate colour luminance from colour information. The conversion from RGB to HSV was done using the OpenCV library method **COLOR_BGR2HSV**.

4.1.5 Client-Server Connection

In order to connect the client application to the server application a form of network node communication needed to be established. This was achieved using socket communication, this process involved first creating a socket object on the server and then binding the host address, the private IP address of the server, and the port number, in this case 7777. The port number 7777 was chosen because it is not one of the reserved ports, reserved ports are from 0 to 1023(system ports) and 49151 and above(dynamic/private ports). The next step was for the server to listen on that port for all incoming connections, this is a synchronous method which blocks and waits until a connection is received, this allowed it to receive data on that port. The client connection is initialized by creating a socket object and then binding the servers public IP address(not private) and the port number 7777. Next a connection attempt is made from the client to the server, if successful the server accepts the connection and returns to the listening state. The client receives the accepted connection response and the socket is then open to send and receive data to and from the client and server. This is achieved utilizing the Socket Python library.

4.1.6 Compression and Serialization

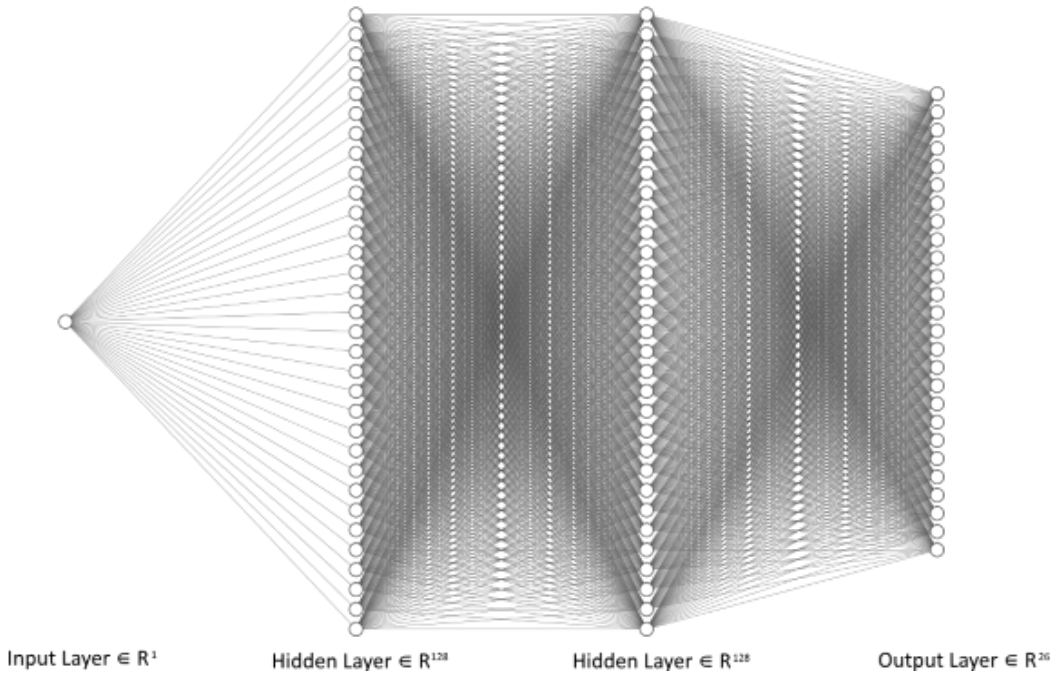
All sent socket data is first compressed using the Blosc compression algorithm, data compression(also known as source-coding or bit-rate reduction) is the process of encoding the bits of data in such a way that it consumes less disk space but still retains all vital information. After the the compression step the data is then serialized using the Pickle Python library, serialization is the process of translating a data structure or an object into a format that can be transmitted over a network. The compressed and serialized data is then transmitted across the network, when the data is received the process defined must be reversed to return the data to it's original state.

4.1.7 Neural Network

An artificial neural network, or ANN, is computing model designed loosely around the model of the human brain. An ANN is not an individual algorithm, it is a framework that facilitates the coupling of many different machine learning algorithms in order to process complex data recognition tasks. The ANN implemented in this project is known as a multi-layer perceptron, MLP.

A perceptron is a single node in the ANN, a single perceptron is capable of preforming linear separation, the greater the complexity of the problem area the more perceptrons that are needed. An ANN is comprised of layers, which are groups of perceptrons, each layer in the ANN applies an algorithm to each node which affects it's weights. Weights are the values applied to the input vector at it traverses the ANN, they essentially control the effect, or weight, that input has on the output.

A supervised ANN is able to "learn" a generalized pattern without being programmed with any task-specific rules, in essence it is an abstraction. A supervised neural network is essentially a form of data mining that infers a function from labeled training data, this function can then be used for mapping new examples. The objective of a supervised ANN is to allow the collection of algorithms to correctly determine the class labels for unseen input data.



As can be seen from the topology diagram above, the artificial neural network is comprised of four layers, one input layer, two hidden layers and an output layer.

The input layer contains only one node, this layer is a Flatten layer, which takes in an image array of size [50,50,1](the pixel height and width). The Flatten layer reduces the dimensions of the input data to one dimension, this flattening is preformed in order to couple information that exists vertically(columns) with the information that exists horizontally(rows).

The first and second hidden layers are Dense layers, this simply means that every node in each layer is connected to every node in the layer before and after. These Dense layers contain 128 perceptron nodes each. The activation function applied to each layer was relu(Rectified Linear Unit)[2], this activation function returns the element-wise max of x and 0 by applying the following rectified function:

$$f(x) = x^+ = max(x, 0)$$

The output layer is also a Dense layer, this layer contains twenty-six perceptron nodes, one for each letter of the alphabet. The activation function applied to this layer was softmax[2], this activation function takes as input, a vector of K real numbers and normalizes them. Normalization is the process of translating values into either a 0 or 1 value.

The final trained weights are saved after training is completed, these were stored in order to repeat the achieved error rate. The weights files are stored with the .h5 extension, a selection of these can be found on the GitHub repository provided.

4.2 Issues Faced

The purpose of this section is to discuss a selection of the challenges faced during the development process of this project, and the steps taken to overcome them.

4.2.1 Graphical User-Interface

The graphical user interface element of the project was initially intended to be developed using Ionic. Ionic is a software development kit for developing hybrid mobile applications. During the development of the mobile application it became apparent that a mobile application was not suitable to display the necessary elements of this project. The Ionic mobile application was rejected due to this limitation, in it's place a desktop application was developed using the Tkinter Python GUI library. The use of this package to develop the GUI aspect of this project was an optimal solution due to the back-end of the already being developed using the Python programming language.

4.2.2 Client-Server Implementation

Initially, the server was developed with all functionality except for the GUI and video capture, which were client side. The data needed to constantly be sent to and from the server, this caused latency issues and reduced the overall user experience. While this was done in an attempt to reduce the users' associated space complexity it resulted in a negative effect. This method was rejected and instead only the artificial neural network was incorporated into the server application.

4.2.3 Antimalware Service Executable Issue

During the course development on the Amazon Web Services server I noticed a steady drop in processing speed. As there is a 1GB memory limit when using the free tier service I had initially suspected it may just be the lack of resources until all process came to a standstill, after checking with task manager a process named "Antimalware Service Executable" was consuming

over 80% of the system memory. This process is coupled with the Windows Defender Service and comes bundled with Windows 10. There is a known issue with this process consuming an exorbitant amount of processing power, so after some parsing of forums I found several potential solutions to this issue. None of the found solutions had any effect on reducing the resources consumed and attempting to fix this issue was costing too much time as the system was overburdened by this issue.

The solution taken was to create a new AWS server instance an then perform all of the solutions found to fix the issue on the new instance as a precautionary measure. The steps taken were to remove all tasks scheduled by Windows Defender, adding the Antimalware Service Executable process to the Windows Defender exclusion list and disabling Windows Defender.

4.2.4 Normalization Issue

On the server, while normalizing the training data portion of the dataset before training an issue was encountered. This issue was perhaps due to the limitations with the Amazon Web Server memory resource, the amount of available RAM was only 1GB. The solution to this problem was to create a Python script to normalize the training and testing data locally on my own computer and then upload them to the server. The server application was then altered to accommodate this change.

4.2.5 Library Version Issues

- when re-starting the server I ran in to many version issues - using Anaconda prompt to install packages - major issue with OpenCV, in order to have the necessary version of OpenCV when using the conda keyword to install packages it downgraded the version of Keras I was using which cause an issue when compiling as it was missing a reference - to fix this I used the PIP, Python Packet Manager, to install OpenCV and conda to re-install the correct version of Keras.

Chapter 5

System Evaluation

The purpose of this section is to evaluate the project goals and discuss the progress made towards them. This section will also analyze the system design and highlight the advantages and limitations with the technologies used.

5.1 Objectives

The objectives for this project were as follows:

- To train an artificial neural network to recognize the American Sign Language alphabet.
- To create a real-time gesture recognition application interface.
- To expand the artificial neural network beyond the ASL alphabet.
- To host the artificial neural network to allow for threaded multiple-user access.

The first objective was to create and train an artificial neural network to recognize the letters of the American Sign Language alphabet. The artificial neural network developed proved to be extremely accurate, correctly identifying over 99% of the testing dataset images. I feel like this objective was met to a high standard due to the accuracy achieved.

The second objective was to create a real-time gesture recognition interface. The Tkinter graphical user interface implemented meets this objective however, the interface is quite lacking in user-friendliness and is quite minimal. I feel like this objective was met but further development is needed.

The third objective was to expand the artificial neural network beyond the ASL alphabet. This objective was not met, this is mainly due to the

revision of the project plan as outlined in section 2.1.3. This aspect however, would be an area for future development of this project.

The final objective was to host the artificial neural network and to allow for threaded multi-user access. This objective was met, the server application can receive a new connection request, the server will then start processing that request on a new thread while the main thread continues to listen on the open port for a new connection. This is limited by the AWS free tier server as resources are limited two simultaneous connections cannot be supported.

5.2 System Design Decisions

The design decisions made on the graphical user interface element are quite limited and are implemented with the sole purpose of demonstrating the functionality of the application. This element needs to be improved upon in order to improve the user experience.

The full HSV(hue saturation value) range can be selected using the slider GUI elements, this however is undesirable as only a selection of the HSV range is viable when extracting skin tone. In order to reduce this a limit must be implemented to reduce the select-able HSV range.

5.3 Limitations and Advantages

The limitations faced with the technology used in this project were mainly centered around the Amazon Web Services server. The limitations of resources when using the AWS free tier option was highlight in sections 4.2.2 and 4.2.3, these limitations had a significant impact on the project plan schedule. In hindsight, this impact should have been foreseen due to the known lack of resources on the AWS free tier option. The other technology limitation faced was the Tkinter graphical user interface library, this limitation was not so much functional, but designed based. Due to the quite dated interface it could hinder the user experience. The advantages of the technologies implemented in this project were the small learning curve associated with both the OpenCV and Keras libraries. The OpenCV library is quite extensive and robust, it's no surprise why this library is so popular and widely utilized in the field of computer vision. The Keras documentation is quite extensive, quite practical and straightforward, complex topics are delivered in a concise and direct manner.

Chapter 6

Conclusion

Due to the limitations with the AWS server the server application is not very reliable when processing multiple user access, although the functionality is present for this. The project presentation element also suffered due to the limitations of the chosen GUI technology.

The project area is much smaller than intended but I feel that this project successfully implemented these technologies to develop an application that showcases their potential applicability to overcome communication barriers.

Bibliography

[1] T. B. Developers, “What is blosc?.”

[2] K. D. Team, “Keras activation functions.”