

HAND TALKER: A FILIPINO SIGN LANGUAGE BASED GAME

A Special Problem

Presented to

the Faculty of the Division of Physical Sciences and Mathematics

College of Arts and Sciences

University of the Philippines Visayas

Miag-ao, Iloilo

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science by

Madriaga, Justin Loise C.

Solano, Julius Caesar A.

Francis Dimzon

Adviser

June 2, 2022

Approval Sheet

The Division of Physical Sciences and Mathematics, College of Arts and
Sciences, University of the Philippines Visayas

certifies that this is the approved version of the following special problem:

HAND TALKER: A FILIPINO SIGN LANGUAGE BASED GAME

Approved by:

Name

Signature

Date

(Adviser)

(Division Chair)

Division of Physical Sciences and Mathematics

College of Arts and Sciences

University of the Philippines Visayas

Declaration

We, Justin Loise Madriaga and Julius Caesar Solano,, hereby certify that this Special Problem, including the pdf file, has been written by us and is the record of work carried out by us. Any significant borrowings have been properly acknowledged and referred.

Name

Signature

Date

Justin Loise Madriaga _____
(Student)

Julius Caesar Solano _____
(Student)

Acknowledgment

Firstly, the researchers would like to express our gratitude to the initial advisor who accepted the proposal Ma'am Christi Florence Cala-or. She was able to initially flesh out the thesis and gave proper remarks and corrections that would let us proceed further towards the research.

Next we would especially thank our current advisor for the thesis, Sir Francis Dimzon. He was able to flesh out even more on our thesis paper and was able to provide helpful tips on how to use certain functions on MATLAB as we were inexperienced at MATLAB beforehand and provided guidelines as to how we should program the application.

Sincerest thanks also go to JPI, especially Raymund, Marco, Tope, Keane, and Jeiel. Apart from them providing companionship, they also contributed in providing the researchers with data for the dataset. Also Raymund thanks for the manuscript that made making this thesis much more smoother to make. Hopefully we'll all meet again in person where we could do our usual bs.

Thanks also goes to one of the researchers old highschool friends, Jackie, Dineil, and Jao. These guys provided the researchers with data for the dataset and also provided support to continue the thesis despite the delay. Thank you for being good friends especially for this long

Also special thanks to John Fuentes, who is also one the last person to provide data to the dataset and also for letting one of the researchers to destress in your house together with Marco and Tope. I was honestly fun playing games and forgetting the thesis stress.

Finally to thank the researchers parents, for providing us with food, shelter, support in these hard times especially with COVID-19 still going. We wouldn't have done further with our thesis and to extent our lives without your love and support.

Abstract

Hand Talker is a video game involving sign language recognition to promote and help users in learning the Filipino Sign Language. This project utilized MATLAB's Deep Learning, Image Processing, Computer Vision, and Bioinformatics Toolbox in obtaining and processing images via Convolutional Neural Network. The dataset used were static Filipino Sign Language images from human gesture(except for letters J, Ñ, NG, and Z) with 10,20,30, and 40 image iterations. The game itself is compromised with a timer, the word, the score, lives left and the camera. How this game works is that the player is given a word from a preset dictionary and within 2 minutes the player has to spell out the word using FSL sign language and you are given 3 lives for the entire game. In the experiments, the default model for sign language detection has a case of overfitting in which new hand gestures were not detected accurately. For future reference, in order to avoid overfitting it is best to include a larger amount of training dataset, have better system specifications, and account for motion detection.

Keywords: Filipino Sign Language, Machine learning, Machine learning approaches, Neural networks

Contents

1	Introduction	1
1.1	Overview of the Current State of Technology	1
1.2	Problem Statement	2
1.3	Research Objectives	3
1.3.1	General Objective	3
1.3.2	Specific Objectives	3
1.4	Scope and Limitations of the Research	3
1.5	Significance of the Research	4
2	Review of Related Literature	6
2.1	Gesture Recognition	6
2.2	Image Processing Algorithms	7

<i>CONTENTS</i>	viii
2.3 Hand Gesture Recognition	8
2.4 Convolutional Neural Network	9
2.5 AlexNet	11
2.6 Transfer Learning	13
2.7 Other Studies	14
3 Research Methodology	17
3.1 Developing the GUI of the System	18
3.2 Developing the Sign Language Detector	20
4 Results and Discussions	24
4.1 Sign Language Detection	24
4.2 Test Data Result	25
4.3 HandTalker Application	26
4.3.1 High Score GUI	27
4.3.2 How to Play GUI	28
4.3.3 Start Game GUI	29
5 Conclusion	30

<i>CONTENTS</i>	ix
5.1 Conclusion	30
5.2 Recommendations	31
6 References	33
A Source Code of the Application	38

List of Figures

2.1	The AlexNet Architecture.	11
2.2	Diagram on how Transfer Learning Works	13
3.1	The Actual game window for HandTalker	19
3.2	Sign Language Format used in the Sign Language Recognition . .	21
3.3	Sample Images that have been undergone Data Augmentation for Training the dataset	22
4.1	The Sign Language letter ‘C’	24
4.2	MainMenu for HandTalker	26
4.3	The HighScore Part of HandTalker	27
4.4	The How to Play Section of HandTalker	28
A.1	Main Program for training the data set	38

A.2	The preproc function which handles the Data Augmentation . . .	39
A.3	Function that resizes all the images into 227 x 227	39
A.4	Part 1 of the main code for the actual game framework of HandTalker	40
A.5	Part 2 of the main code for the actual game framework of HandTalker	41
A.6	Part 3 of the main code for the actual game framework of HandTalker	42
A.7	Part 4 of the main code for the actual game framework of HandTalker	42

List of Tables

4.1	The Validation Accuracy of Trained Datasets with each iteration .	25
-----	---	----

Chapter 1

Introduction

1.1 Overview of the Current State of Technology

One of the challenges the deaf and mute people face is language comprehension compared to ones who can listen properly. That being said, this leads to these disabled people restricted from social interactions. Solution methods such as learning sign language are devised so that these disabled people have a chance of creating healthy social relationships with other people. For the case of the Philippines according to RA 11106 the Filipino Sign Language has become the standard for deaf and mute communication (NCDA, 2018).

In a national survey in the Philippines of 2275 adults and children an estimated 15% in the Philippine population has hearing problems. Within the said survey there were 7.5% in children, 14.7% in adults between 18 and 65 years and 49.1% in adults aged 65 years above that have hearing problems(hear-it, 2021).

Learning sign languages takes time and becomes more challenging the older you become. However, with sign language the only time to only practice sign language is only limited to a few people that have difficulty hearing or speaking which can limit how quickly a person can master sign language. Another challenge for learning is that the average speed of communication in a social setting can get overwhelming which also includes basic non verbal communication like body language and proper eye contact.

According to Rios-Soria et al(2013) hand recognition has become a popular topic in computer vision. The computer is able to detect user input without the actually touching the system using specific hand patterns, filters and convex-hull detection in which in some cases only using a standard webcam without any special markers.

In this study, a video game involving sign language recognition is developed in order to promote and help users in learning the Filipino Sign Language.

1.2 Problem Statement

Deaf and mute people tend to have difficulties understanding language, especially against people that can talk and hear properly. The same is true for the abled people trying to understand other people who are deaf and mute people. Sign Language is devised in order to help these people to understand each other, however learning sign languages can be difficult.

1.3 Research Objectives

1.3.1 General Objective

The general objective of the study is to create an educational video game based on sign languages using Computer Vision to help people understand and learn about the Filipino Sign Language.

1.3.2 Specific Objectives

- To develop a system that will detect the sign language using the user's hands
- To create a dataset that contains the Filipino Sign Language
- To check the accuracy of the sign language detection from the system, especially with other factors that can affect the accuracy such as the background, the skin color of the user, and the size of the hands of the user.
- To evaluate the gameplay of the educational video game based on sign languages.

1.4 Scope and Limitations of the Research

This video game application was developed using MATLAB r2022 using Windows 10. It used MATLAB toolboxes such as Deep Learning, Image Processing, Computer Vision, and Bioinformatics Toolboxes. The system is made as a time trial video game where the user must spell out a random word presented on the

screen by performing sign languages inside a video camera within a time limit. The system also included a menu where the user can type out a word and a sign language representation would be displayed on screen.

The program might not run as well for slower computers, especially when the system the users use is worse compared to the one used in this study. This program might also not be applicable to users having finger stiffness or trigger finger which will encounter difficulties in performing the sign languages. The people involved for the dataset are not deaf or mute and are also inexperienced with sign languages in general which could affect the quality of the dataset for this research.

The system only incorporates the Filipino Sign Language Alphabet. The system was also not able to incorporate more complex sign gestures that usually involve moving sign gestures (like the letter J or letter Z) or complex gestures involving more than just one hand. The system for it to work would require at least a functional webcam for capturing purposes. The research was conducted at Malipayon Village, Roxas City, Capiz during the school year 2020-21.

1.5 Significance of the Research

The results of this study will benefit the following:

STUDENTS. More specifically, students in computer courses and language courses will benefit with the results of this research. The language students will benefit in further improving their knowledge in sign language. Computer related students

will benefit from hand gesture recognition and the coding behind it.

PEOPLE WITH DISABILITIES. People who are deaf, mute or with hearing disabilities can greatly benefit from the application as it helps them understand better on how to communicate without speech. People related to these disabled people would also benefit from this as it would help them communicate better to their respective disabled friend or relative.

LANGUAGE TEACHERS. These teachers can benefit as an aide to teach their students on how to use proper sign language.

COMPUTER DEVELOPERS. These experts can benefit from this study by further improving the application into a more optimized and more engaging video game.

FUTURE RESEARCHERS. This study will provide baseline data needed for future research and studies related to this one.

Chapter 2

Review of Related Literature

2.1 Gesture Recognition

According to Rui Ma (2021), gesture recognition is a computing process that uses human gestures, ranging from hand gestures, facial expressions to body positions, in which the computer tries to recognize and interpret with the use of various mathematical algorithms. Gesture Recognition is divided into two categories, Contact recognition where the technology involves the user wearing sensors around the human body in order to realize the gesture recognition and the non contact recognition technology which uses computer vision. The application for this study will utilize the noncontact Recognition Technology using a webcam. Generally the vision based gesture recognition technology uses camera and motion sensors to track user movements and translate them in real time. In some newer cameras and programs the recognition would take into account the depth data as well which can help in gesture tracking.

Human body recognition is an important research for computer vision. Its ultimate purpose is to use a person's overall or partial parts of the body as input to get output data such as, outline of the human body, the positions and orientation of body parts and positions of human joint points or category of parts. Research methods of gesture recognition cover most theories in the field of computer vision such as pattern recognition, machine learning, artificial intelligence, image graphics and statistics .

2.2 Image Processing Algorithms

According to Huu and Ngoc (2021), there have been multiple image processing algorithms that have been developed in target detection and recognition. These algorithms are divided into two categories, which are Machine Learning and Deep Learning Techniques.

Machine Learning techniques are usually used with basic feature extraction methods from original data and then combining them. Support Vector Machine, decision tree and nearest-neighbors, to train identity models. Some examples of extraction techniques for object detection are Viola-Jones target detection, Scale-invariant feature transform (SIFT), Histogram of oriented gradients (HOG). Viola-Jones target detection is utilized mainly for frontal faces where an image is converted into grayscale to reduce data to process and the algorithm tries to find a face within the image. SIFT is used to match features of images even if the image has different scalings or rotations of the feature. HOG is more focused on the structure or shape of an object and is done by extracting the gradient and

orientation of the edges .

Deep Learning Techniques are the more common techniques used in modern computer vision. These Deep Learning Techniques often use multilayered convolutional neural networks (CNN) training on labeled datasets. It is used to reduce data and calculations to the most relevant set and this set is then compared against provided known data to identify or classify the data input. In this study CNN with the library of OpenCV will be used, as OpenCV comes with a lot of pre-defined functions. Deep Learning architecture have been applied in object detection and recognition include, AlexNet, GoogleNet, VGGNet, ResNet, Xception, ResNeXt-50 and other more (Saha, 2018).

2.3 Hand Gesture Recognition

According to Kumar (2018), hand gesture recognition uses different methods from different researchers for various fields. These various hand gestures were done using glove based approaches and vision based approaches.

Glove-based approaches require users to wear a sensor glove or colored glove to ease the segmentation process when detecting the hand gesture. However it has the main drawback of requiring a specified glove for the user to wear.

Vision based approaches use cameras to detect hand gestures. This technique is much simpler because there is no need for the user to wear any extra hardware. However, it is considered to be less accurate related to image processing algorithms and yet to be fully optimized.

There are also two different approaches to vision based sign language recognition which are, the 3D model based and appearance based approaches. 3D model-based approaches use 3D information of key elements of the hand parts, either with volumetric models, skeletal models or even the combination of two. Appearance-based systems use 2D images as inputs, usually from videos/images, and do not use a spatial representation of the body. The parameters of appearance based systems are derived directly from images or videos using a template database like deformable 2D templates of the human body, particularly that of the hands.

2.4 Convolutional Neural Network

A Convolutional Neural Network is a Deep Learning Algorithm which can take in an input image, assign importance to various aspects/objects in the and be able to differentiate one from the other. The architecture of a CNN is similar to the connectivity pattern of neurons of the human brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. A group of such fields overlap to cover the entire visual area. The main role of CNN is to reduce images into a form where the important features are retained, which is important for getting good predictions (Saha, 2018).

The convolutional layer is the first layer in which input data is usually processed when put into the CNN. The convolution process requires three elements which is the image to convolve, the kernel and the convolution stride parameter. For example, an image of 5x5 matrix as input; the kernel which is a smaller image

has a 3x3 matrix. Starting from a corner of the image, the filter is mapped to a region of the image called a receptive field. Each pixel in the image is considered a neuron and its values are multiplied by the convolution filter neurons, in which its resulting values are to be added and a single value is computed and stored in the feature map. After convolution is performed, the kernel or filter moves to the right within the stride parameter where convolution occurs again for that portion of the image until it parses the complete width of the input image. Afterwards, the filter hops down by the same stride value from the left and repeats the process until the entire image is traversed. The more kernels used in the model the more feature maps are produced which potentially increases recognized features. Training a CNN focuses on learning the values of neurons of convolution filters, which are called weights. At the beginning the weights are set to random and after applying backpropagation algorithms the system would update these values to fit the training data (Lopez, 2017).

The Rectified Linear Activation function or ReLU is a piecewise linear function that will output the input directly if positive or otherwise output zero. This function has become a default activation function for many neural networks as it makes training easier and achieves better performance. In CNN this is usually inline with the convolutional layer in order to speed up the process while also retaining performance (Brownlee, 2019).

The feature maps created from the convolution layer is then taken to the pooling layer where it reduces the dimensionality of the image while its dominant features are retained. There are two types of pooling which are the Max and Average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel, while Average pooling returns its average of all the val-

ues. Max pooling additionally has noise suppression where it would discard noisy activations (Lopez, 2017).

The convolutional layer and pooling layer together would add up as a single layer of a CNN and multiple layers can be added for increased complexity at the cost of computational power (Saha, 2018).

After all these processes, flattening is done unto the output and is fed into a regular neural network or otherwise known as a fully connected layer for classification. This layer produces the predictions of the input image and thus the number of output neurons is equal to the number of classes (Lopez, 2017).

2.5 AlexNet

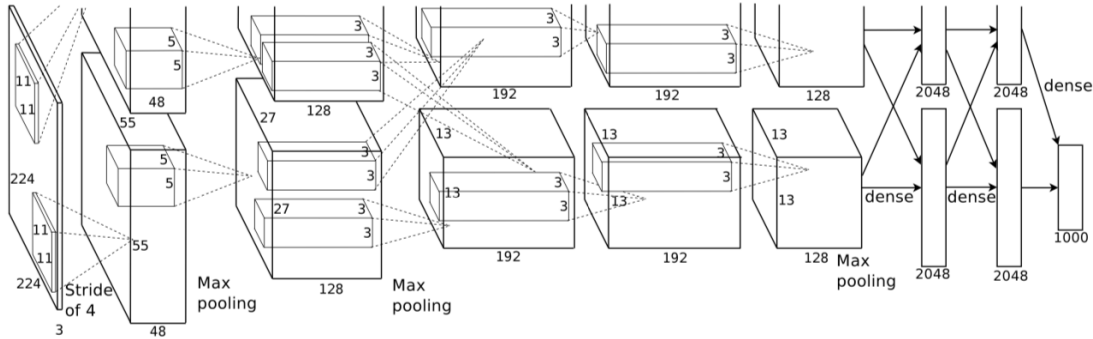


Figure 2.1: The AlexNet Architecture.

AlexNet is an architecture model developed by Alex Krizhevsky and his colleagues in a research paper named Image Classification with Deep Convolution Neural Network which won the Imagenet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012.

The architecture itself consists of eight layers with learnable parameters. Its model consists of five convolutional layers and three fully connected layers. It uses Rectified Linear Units (ReLU) which helps in improving the training time by around six times and is used in all of the convolutional layers and fully connected layers except the output layer. The input to this model is the images with sizes $227 \times 227 \times 3$.

AlexNet also supports Multiple GPUs; back then the norm for training is around three gigabytes of memory, which is bad for the Imagenet training set having 1.2 million images. AlexNet however allows for multi-GPU training by splitting the work for training on multiple GPUs which significantly cuts down training time.

Overfitting is when a Machine Learning Model tries to fit exactly towards all the data points in a given dataset. These might affect accuracy, especially when trying to use an overfitted model towards the test dataset. The authors for AlexNet used two methods to reduce overfitting.

The first is Data Augmentation. In their study the authors used label preserving transformations to make the images look more varied. They generated image translations and horizontal reflections. They did this by extracting random 224×224 patches from the 256×256 images and training the network on the extracted patches. They also alter the intensities of the RGB channels in training images using Principle Component Analysis (PCA) which reduced the top-1 error rate by more than 1%.

The other method used is by using Dropout. Combining the predictions of many different models would take too much time and be too expensive especially on big neural networks which already take several days to train. Dropout is a tech-

nique that basically sets to zero on some of the output of some neurons with the probability of fifty percent. This means that for every input presented the neural network would sample a different architecture sharing the same weights to other architectures. Dropout Layers are used in the first two Fully Connected Layers in the AlexNet Model.

In this study we used the AlexNet CNN for the Sign Language Detection using Transfer Learning.

2.6 Transfer Learning

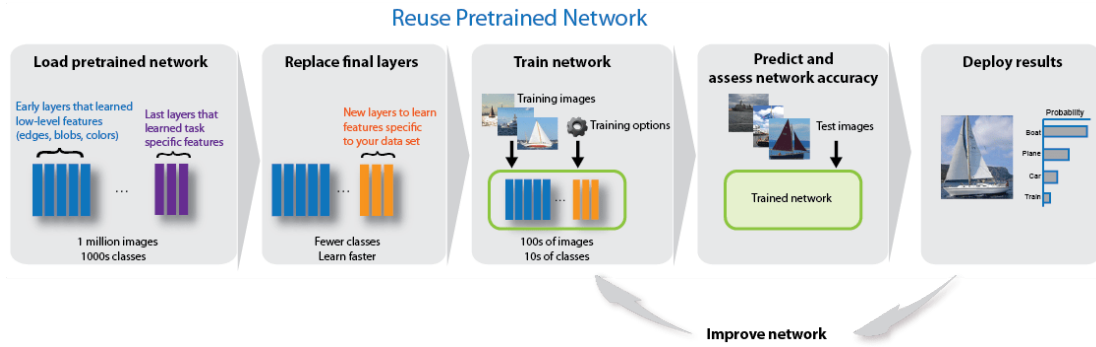


Figure 2.2: Diagram on how Transfer Learning Works

Transfer Learning is the process in Machine Learning that reuses a pre-trained model on a new problem. Fine tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch. You can quickly transfer learned features to a new task using smaller number of training images.

In this study we used AlexNet as the pre trained CNN and how we performed Transfer Learning is by replacing the last three layers of AlexNet and finetuned

it in such a way that it would fit better for the sign language detection.

2.7 Other Studies

A hand gesture recognition algorithm was made by Huu and Ngoc(2021) for control of a robotic system. The algorithm in the study used support vector machines(SVM) and histogram of oriented gradient and applied these techniques of applying problem related to control robotic systems. The aim for the study is to detect gestures with real-time processing speed, minimize interference, and reduce the ability to capture unintentional gestures. The gestures in the study involve turning on and off the switch and increasing and decreasing the volume of the device. Results show that the algorithm is up to 99% accuracy with a 70-millisecond execution time per frame that is suitable for industrial applications. The results of this study were used like the implementation of CNN and how the algorithm is made.

There was also another study by Kumar (2018) involving the recognition and conversion of 26 hand gestures in Indian sign language into text using MATLAB. The proposed system involved four modules such as: pre-processing and hand segmentation, feature extraction, sign recognition and sign to text. Some of the features were extracted using Eigenvalues and Eigenvectors which are used in recognition. The Linear Discriminant Analysis algorithm was used for gesture recognition and said gesture is translated into text and voice format. The methodologies used in the study, especially the preprocessing of the image, are to be used for the system.

A hand gesture recognition system was made using Python and OpenCV was

developed by Ismail et al (2021). The system detects the gesture of the hand in real time video and is classified with certain areas of interest. The video in which a hand is to be detected is to be analysed via a Haar-cascade classifier to detect the gesture of the hand before the image processing is done. The detection of the hand was done using the theories of the Region of Interest. Various gestures such as the hand showing numbers from 1 to 5, gun, stop and ok gestures were tested and showed that the system accurately detected these gestures. However, there were limitations such as the lightning environmental changes being influential to the segmentation process(background removal). The idea of using the region of Interest is used similarly to this study where only a specific area of the camera is only used when attempting to capture the hand.

A study about an educational video game by Carter Daniel(2017) was built around paleontology arts was made in order to prove that educational video games would serve significant educational purposes. It was also mentioned in this study that video games such as Zoo Tycoon 2 has helped to provide educational tools in the form of encyclopedias and focus on preserving animal life and endangered species. Minecraft would also prove its educational purposes as well, especially towards the disabled. Puzzle games like The Talos Principle help students and players stay engaged and motivated to continue. Finally, PopCap Games' Typer Shark! Deluxe, which is the main basis for the system's gameplay is a relatively simple game which is focused around typing certain letter patterns in the keyboard as quickly as possible which helps in general typing speed when using the keyboard. Although they cannot ultimately replace traditional teaching techniques and that these games are more of a specialized learning rather than general, these games are still useful in providing a fun alternative to revision exercise as to keep students

active in engaging in learning.

A study called Signapse by Alban Joseph and his team is a free open source software that helps everyday people learn sign language for free. It is currently configured to teach ASL or American Sign Language. It was built using OpenCV, C++, Raspberry Pi, TensorFlow and more. The model used MobileNetv2 as the pretrained model for layer transfer and used ImageNet as the initial dataset then trained more using a free ASL dataset on Kaggle containing over 87,000 images by Akash.

With all these studies considered are inspirations in that it convinces the researcher that a Video Game based around sign language is possible. The use of Convolutional Neural Network were used mainly for it being easy to code around and the Region of Interest helped too in minimizing the amount of pixels the CNN takes into account. The video game Typer Shark Deluxe was the main inspiration for the researcher in what the design of the game would be like.

Chapter 3

Research Methodology

The Operating System the system was made with is Microsoft Windows 10 Pro and a x64 based PC. Only one processor is installed which is the Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz with a 7.88GB RAM. The Graphics Card used on the training part of the system which will be mentioned later is the NVIDIA GeForce GTX 960M with a Driver version 497.09.

The Traditional Approach or Waterfall model of software development was used in making the system. The waterfall model is compromised with stages starting with requirements, design, implementation, verification and testing and deployment and maintenance. The computer system with the specs mentioned above are the requirements to be used for the system. The design of the game is GUI(Graphical User Interface) based where the user mouseovers through the menus, play, high-score, tutorial, and exit, which will be discussed later. The gameplay design to be decided will be a game where you spell out the words shown in the screen but using sign language. The coding/implementation will be mainly done using MAT-

LAB r2022 using add ons such as Deep Learning, Image Processing, Computer Vision, and Bioinformatics Toolboxes.

There were four training iterations with the first base iteration having a dataset of 301 images per letters for training and test dataset of 10 images per letters for testing. The second having all the images from the first iteration totalling 311 images for training and 20 images per letters for testing. The third having all the images from the second iteration totalling for 331 images per letters for training and 30 images for testing. And the fourth having all images from the third iteration totalling 361 images for training and 40 images for testing. The accuracy of the trained datasets from each iteration was then recorded.

Afterwards the GUI and the game itself was made around the sign detection system. If there are errors or bugs then maintenance is applied to the program.

3.1 Developing the GUI of the System

The Main Menu, which is what the user should see when opening the program. It is compromised with the buttons, Start Game, Tutorial, High Score, and Exit Game.

Clicking the the How to Play button opens up the Tutorial window which explains how to play the game and 28 icons with each representing the letters of the filipino alphabet in its FSL notation. How the camera detects the sign language will be explained later in the “Developing the sign language detection” part of this study.

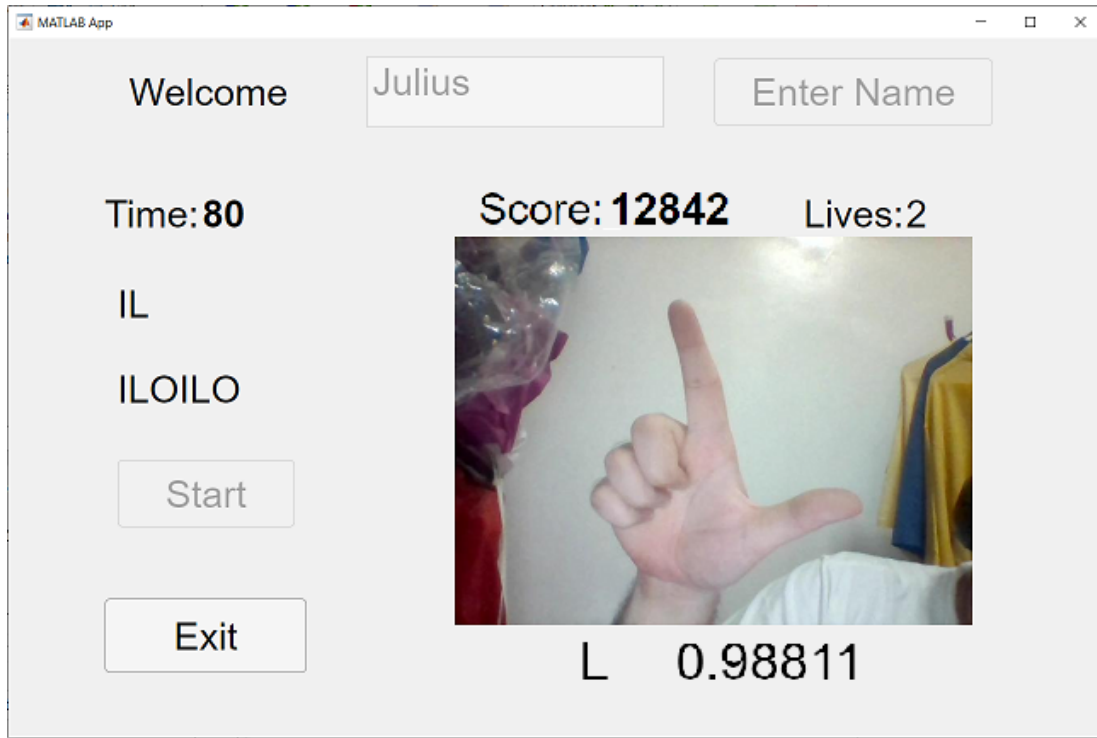


Figure 3.1: The Actual game window for HandTalker

Clicking the Start Game puts the player into the Actual game window as shown in Figure 3.1. The game itself is compromised with a timer, the word, the score, lives left and the camera. How this game works is that the player is given a word from a preset dictionary and within 2 minutes (which is displayed by the timer) the player has to spell out the word using FSL sign language. You are given 3 lives in the entire game.

Points are added to the player whenever the user successfully spelt out the word based on the number of letters the word has and the time it took for the player to spell the word. The player gets points per letter which is affected by the accuracy in by which the sign language was detected which is calculated by the accuracy multiplied by one hundred. More points are added when the user successfully

completes the word within a certain time. The score is added by three hundred if the player spells the word within sixty seconds, two hundred if within thirty seconds, and otherwise just one hundred from just simply completing a word. If you were not able to spell the word within the time given you will lose a life. A new word would then be shown to the game and the user has to spell it out again with FSL.

The game will end once the player has run out of lives. If your current score beats any of the scores in the high score list it will delete the lowest score off the high score and add the current score to its appropriate standing in the list. You will also be prompted to type in the name of the player if you get a highscore. After all this the game ends and will be sent back to the main menu screen.

The Highscore menu should display the local highest scores in this game.

By default, the training data from this research will be used as the classifier for the game.

3.2 Developing the Sign Language Detector

The Sign Language Detector was made using MATLAB r2022 using add-ons such as Deep Learning, Image Processing, Computer Vision, and Bioinformatics Toolboxes. The detector will use CNN in recognizing a hand gesture resembling a sign language. The Sign Language Format to be used for the Sign Language Recognition portion is shown in Figure 3.2 below.

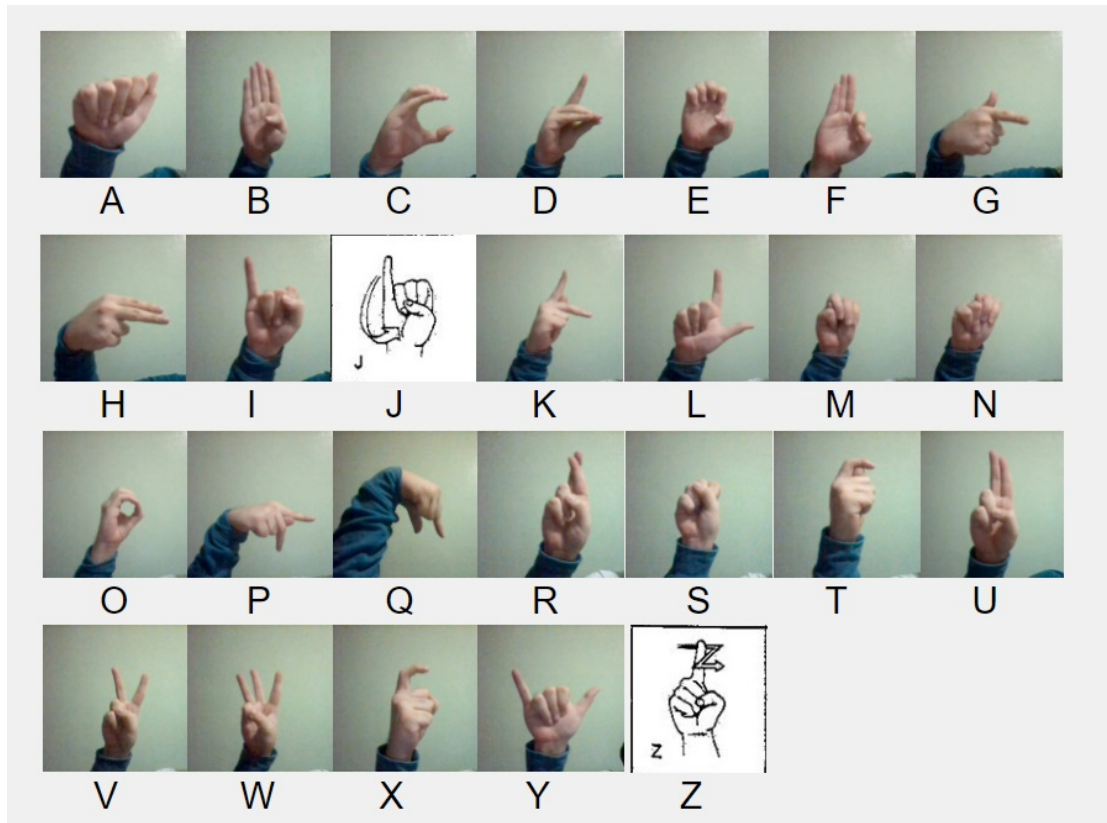


Figure 3.2: Sign Language Format used in the Sign Language Recognition

A custom dataset for CNN training was made for the sign language detector. Using a live feed from the camera in which every frame a hand is shown in the Region of Interest or ROI will be screenshotted and saved into a directory in two folders for train and test, in which each folder is to contain more folders in which they represent each of the alphabet. Background Subtraction is also to be used in isolating the background from the image of the user's hand to be detected by the camera.

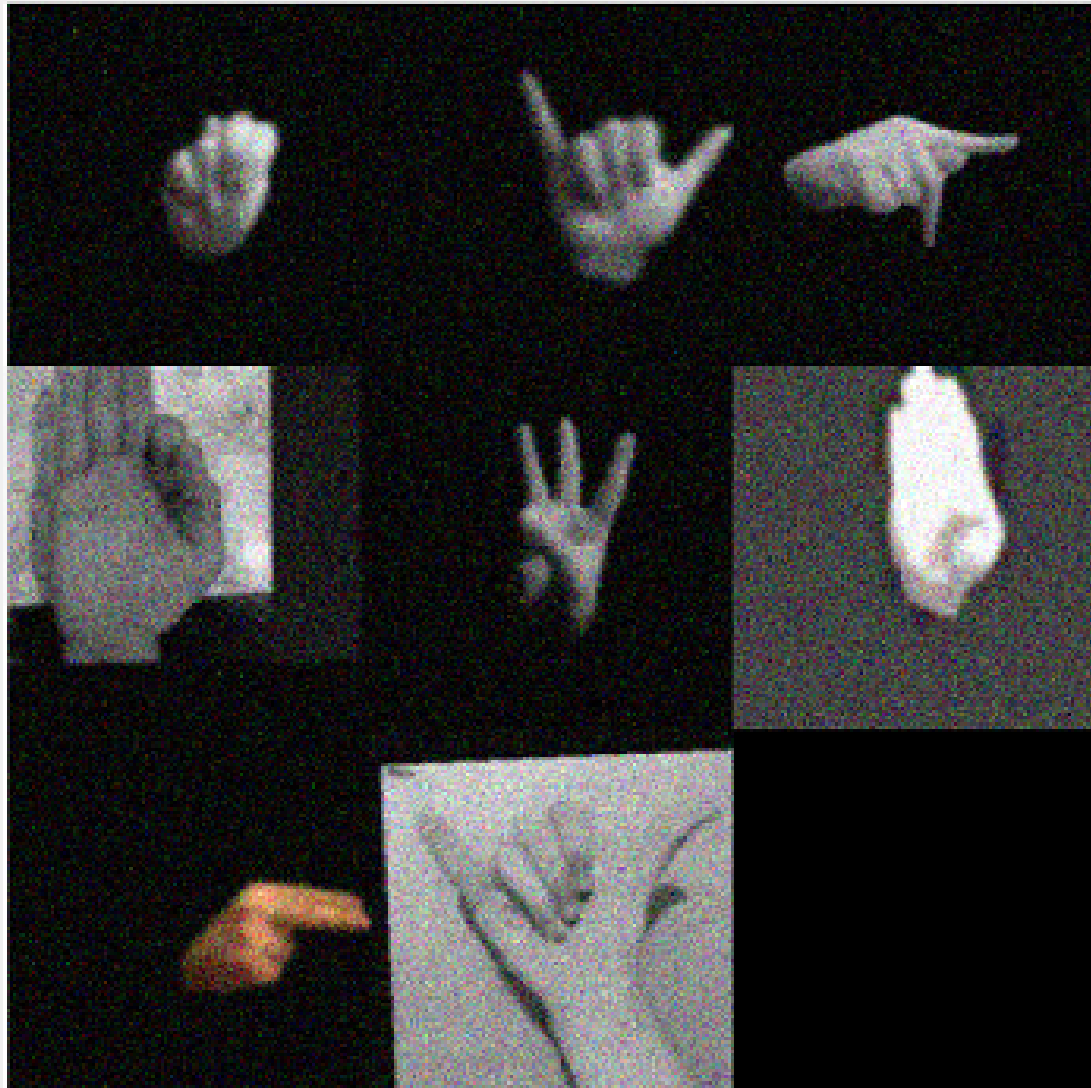


Figure 3.3: Sample Images that have been undergone Data Augmentation for Training the dataset

After the dataset has been created, more preprocessing for the training of the images such as greyscaling, rotating, contrast alteration, brightness alteration, putting gaussian noise, and saturation alteration are applied to reduce the chances of overfitting. Transfer Learning using the pretrained AlexNet is used to train the preprocessed images. Each of these images are to have a 227-by-227 size as

AlexNet uses that size for training.

An SGD optimizer is used when compiling the model. Training the models for each iteration lasted around 1 hr and 30 mins, 2 hrs, 3 hrs and 15 mins, and 6 hrs respectively. After training proceeds to the prediction process. A box in the live feed should be used where the hand should be detected. When the foreground detected from background subtraction is detected as a hand gesture, image segmentation occurs. After extracting the general features of the input the system should try and classify which of these features matches the most to the gestures based on the trained dataset.

Chapter 4

Results and Discussions

4.1 Sign Language Detection



Figure 4.1: The Sign Language letter 'C'

Early Testings when implementing the Sign Language Detection show that the hand should have backgrounds not the same color as the hand as it causes incorrect classifications.

For example Figure 4.1 shows the Sign Language letter ‘C’ however the detector instead classifies it as letter ‘B’. The fix was to instead avoid using backgrounds that are of a similar color to skin.

4.2 Test Data Result

Iteration	Validation Accuracy
1st(10 test images per letter)	25.83%
2nd(20 test images per letter)	21.67%
3rd(30 test images per letter)	23.41%
4th(40 test images per letter)	34.06%

Table 4.1: The Validation Accuracy of Trained Datasets with each iteration

With more testing, the trained dataset is accurate to classify its own training dataset. However, when trying to classify them to new test data the classifier seems to inaccurately classify these images. The case of Overfitting seems to be in play with the trained model. Even with adding more images to the training dataset per iteration, the Validation Accuracy does not seem to increase until the 4th iteration where the validation accuracy of the model improved by around 8.23% compared to the iteration with the second best Validation accuracy. The model seems to mostly bias its classification towards the letters A, F, C, E, and Y. Another possible reason for the model to inaccurately classify the images is because of how similar looking some of the letters are to the detector like how A,

S, M and N look incredibly similar, Y, L, and K look similar and how C and O might look identical. It is also possible that the different hand structures of the datasets for both test and training may have affected the accuracy in sign language detection. Finally it may also be possible that a lack of images for training may still be the case for overfitting.

4.3 HandTalker Application

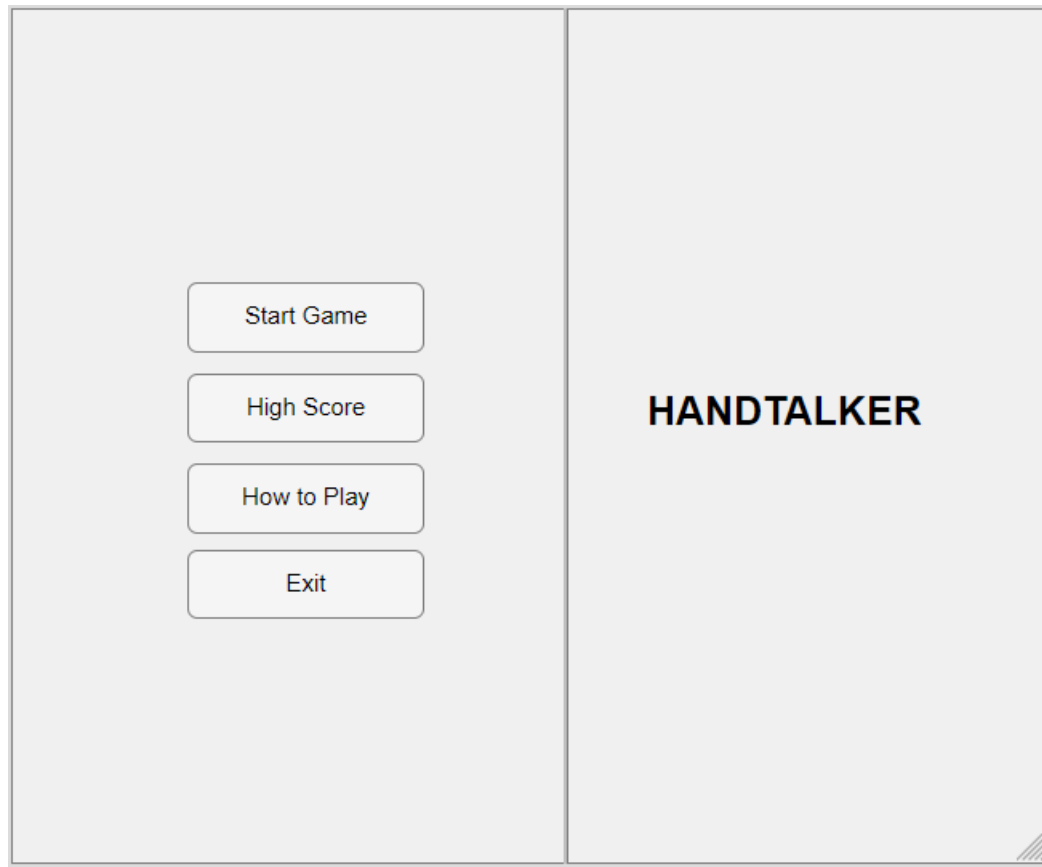
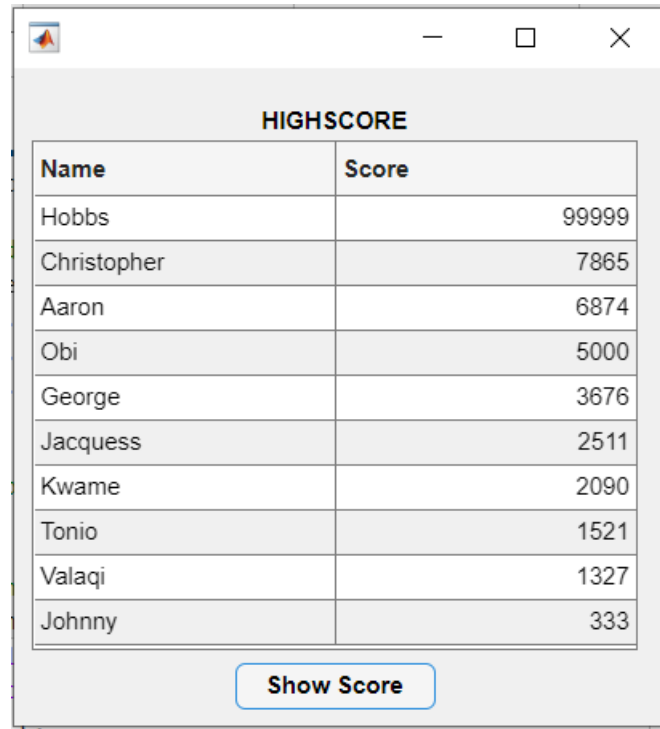


Figure 4.2: MainMenu for HandTalker

The HandTalker Application was made using the MATLAB App Designer which includes the GUI and the main code for the game itself. When starting the application the user is put into the main menu screen shown in Figure 4.2. The user can choose Start Game, High Score, How to Play, and Exit.

4.3.1 High Score GUI



HIGHSCORE	
Name	Score
Hobbs	99999
Christopher	7865
Aaron	6874
Obi	5000
George	3676
Jacquess	2511
Kwame	2090
Tonio	1521
Valaqi	1327
Johnny	333

Show Score

Figure 4.3: The HighScore Part of HandTalker

Figure 4.3 shows the implementation of the High Score GUI for HandTalker which opens up when the user clicks the High Score button in the main menu screen. The High Score GUI displays the top ten players and their respective score when using this application. The way the high score is access is through a table in a.xlsx file which changes whenever a new player has managed to beat a high score

in this application.

4.3.2 How to Play GUI

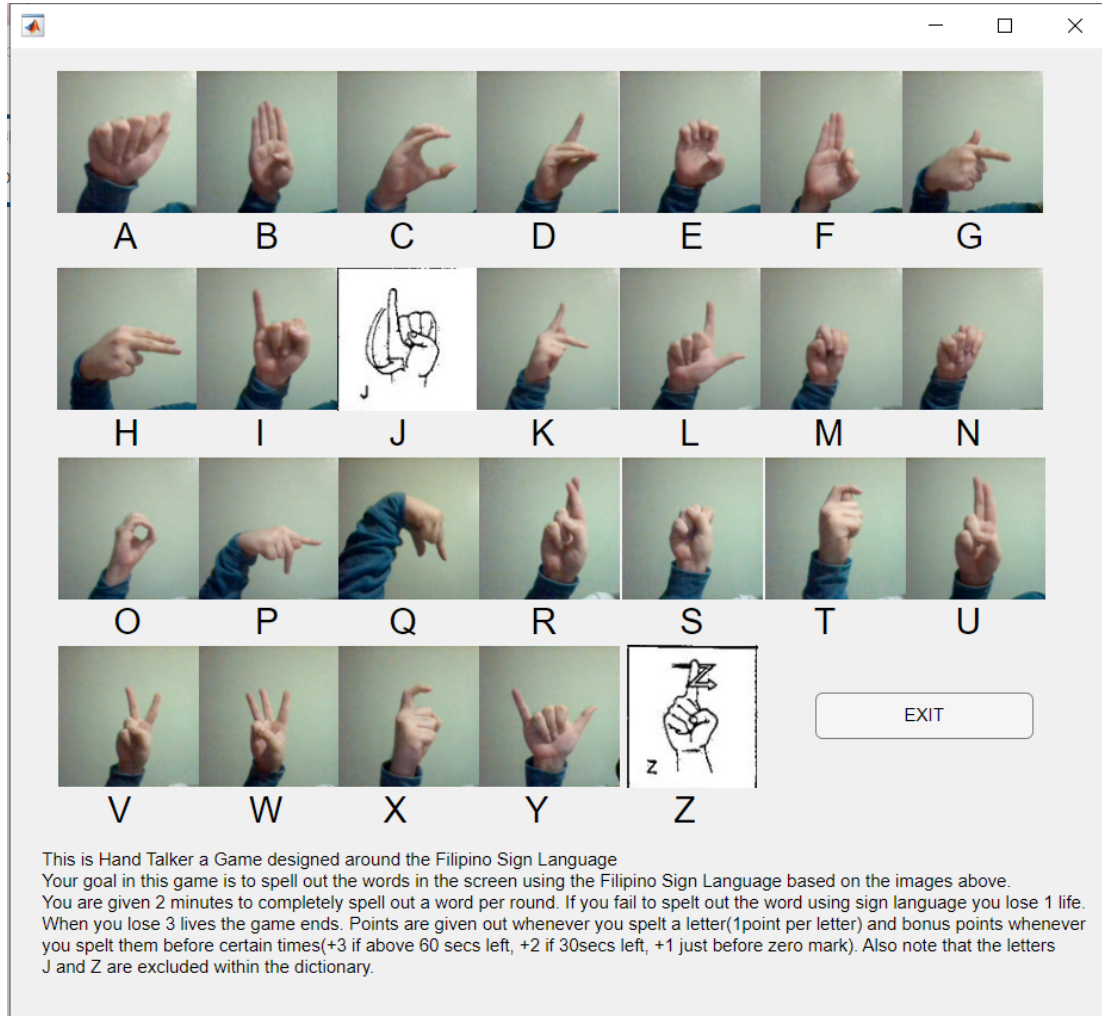


Figure 4.4: The How to Play Section of HandTalker

Figure 4.4 shows the implementation of the How to Play GUI for HandTalker which opens up when the user clicks the How to Play button in the main menu screen. The How to Play GUI displays 26 images which tells the user what Filipino

sign language a particular letter represents then and details on how to play the game. The images are stored in a folder which are accessed by the application to load.

4.3.3 Start Game GUI

Figure 3.1 shows the implementation of the Start Game GUI for HandTalker which opens up when the user clicks the Start Game button in the main menu screen. The Start Game GUI is where the main game for the application starts and is also where the sign language detection model is applied. The game itself is compromised with a timer, the word, the score, lives left and the camera. A dictionary text file is also provided containing up to at least 1000 Filipino words where the application choose one of these words randomly per round and make the user spell out the word displayed in the game. Testing the game itself as the model itself has already been suspected of overfitting the gameplay for the application is sometimes very inaccurate. It would take sometimes too long for the detector within the application to detect the right letters in the word. Sometimes it would simply detect some letters even though it is incorrect. Another important note too is that the model will become even more inaccurate if the face of the user is visible within the camera.

Chapter 5

Conclusion

5.1 Conclusion

This project aimed to develop an educational video game software that involves learning sign language involving Computer Vision. This was done using MATLAB App Designer and by using Transfer Learning using AlexNet as the pre-trained model. While the application can detect sign languages it would only be accurate if the user already has his own hands as part of the dataset.

In conclusion the sign language application by default is not by any means too accurate, requiring the user to input his own sign language images in order to make any accurate detection. The default model for sign language detection has a case of overfitting which disallows accurate detection from new hands to be detected properly. Future students attempting to perform a study similar to this may have some use for it.

5.2 Recommendations

It is very recommended to have better improvements on this model for sign language detection. As the model suffers from overfitting the improvements that could be applied could be ways to avoid overfitting like adding more data to the training set similar to the ASL dataset from Akash which had 87000 total images. Another improvement to the sign language detection could be by using joint based hand gesture detection so that it could more accurately distinguish sign languages especially on some sign languages that look similar to each other.

That said it is also recommended that future researchers attempting to recreate this should have significantly better system specifications compared to the one in this study as currently doing this study took a toll on the system where the model was developed and rendered the training of the model taking longer than usual to complete.

Another recommendation would be to include actual people who are deaf or mute as signers for the dataset or testers. From the Signapse application, there was a comment mentioning that this application has been showing incorrect hand signs for many classifiers especially for T,G,P,Q, and J. There was also an instance in the application where the user signs D with just their pinky finger appears to pass the test. Direct input from an actual deaf person is recommend as they are likely going to be the most experienced when it comes to sign languages and to avoid deaf/non-signer bias.

Finally the sign language detection should be improved in such a way the even motion based hand gestures such as the letter J and letter Z can be incorporated

in the sign detection.

Chapter 6

References

- Akash. (2018). *ASL alphabet*. Retrieved from <https://www.kaggle.com/datasets/grassknoted/asl-alphabet> (Online; accessed 13-May-2022)
- Alom, M. Z., Taha, T. M., Yakopcic, C., & Westberg, S. (2019). *A State-of-the-Art Survey on Deep Learning Theory and Architectures*. ResearchGate. Retrieved from https://www.researchgate.net/publication/331540139_A_State-of-the-Art_Survey_on_Deep_Learning_Theory_and_Architectures (Online; accessed 10-December-2021)
- Berke, J. (2020). *How hard is it to learn sign language?* VeryWell Health. Retrieved from <https://www.verywellhealth.com/challenges-of-learning-sign-language-1049296> (Online; accessed 1-November-2021)
- Bhavana, D., Kumarb, K. K., Chandra, M. B., Bhargav, P. S. K., Sanjana, D. J., & Gopi, G. M. (2021). *Hand Sign Recognition using CNN*. International Journal of Performance Analysis in Sport. Retrieved from <http://www.ijpe-online.com/article/2021/0973-1318/>

- 0973-1318-17-3-314.shtml (Online; accessed 15-November-2021)
- Brownlee, J. (2019). *A gentle introduction to the rectified linear unit (ReLU)*. Machine Learning Mastery. Retrieved from <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/> ([Online; accessed 14-December-2021])
- Burton, M. (2013). *Evaluation of sign language learning tools: Understanding features for improved collaboration and communication between a parent and a child..* Iowa State University Capstones, Theses and Dissertations. Retrieved from <https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=4127&context=etd>. ([Online; accessed 1-November-2021])
- Carter, D. (2017). *Can video games technology be used for educational purposes?*. Masters thesis, University of Huddersfield. Retrieved from https://eprints.hud.ac.uk/id/eprint/31582/1/_nas01_librhome_librsh3_Desktop_FINAL\%20THESIS.pdf ([Online; accessed 14-December-2021])
- Huu, P. N., & Ngoc, T. P. (2021). *Hand Gesture Recognition Algorithm Using SVM and HOG Model for Control of Robotic System*. Journal of Robotics, vol. 2021, Article ID 3986497, 13 pages. Retrieved from <https://web.archive.org/web/20211204093706/https://doi.org/10.1155/2021/3986497> ([Online; accessed 15-November-2021])
- Joseph, A., Gardiner, R., Frew, A., & Russell, L. (2021). *Signapse*. GitHub. Retrieved from <https://github.com/albanjoseph/Signapse> ([Online; accessed 13-May-2022])
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Clas-*

- sification with Deep Convolutional Neural Networks*. NeurIPS Proceedings. Retrieved from <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf> ([Online; accessed 5-May-2022])
- Kumar, M. (2018). *Conversion of Sign Language into Text*. International Journal of Applied Engineering Research, 13(9), 7154–7161. Retrieved from https://www.ripublication.com/ijaer18/ijaerv13n9_90.pdf. ([Online; accessed 5-November-2021])
- Lopez, I. V. (2017). *Hand Gesture Recognition for Sign Language Transcription*. Boise State University. Retrieved from <https://scholarworks.boisestate.edu/cgi/viewcontent.cgi?article=2322&context=td>. ([Online; accessed 10-December-2021])
- Ma, R., Zhang, Z., & Chen, E. (2021). *Human motion gesture recognition based on Computer Vision*. Complexity. Retrieved from <https://www.hindawi.com/journals/complexity/2021/6679746/> ([Online; accessed 15-November-2021])
- MATLAB. (2022). *version 9.12.0.1884302 (R2022a)*. Natick, Massachusetts: The MathWorks Inc.
- Nearly one in six in the Philippines has serious hearing problems*. (2021). hear-it. Retrieved from <https://www.hear-it.org/nearly-one-six-philippines-has-serious-hearing-problems> ([Online; accessed 1-November-2021])
- Patel, K. (2020). *Image feature extraction: Traditional and Deep Learning Techniques*. Medium. Retrieved from <https://towardsdatascience.com/image-feature-extraction-traditional-and-deep-learning-techniques-ccc059195d04> ([Online; accessed 5-November-2021])

- Pinto, R. F., Borges, C. D. B., Almeida, A. M. A., & Paula, I. C. (2019). *Static hand gesture recognition based on convolutional neural networks*. Journal of Electrical and Computer Engineering. Retrieved from <https://www.hindawi.com/journals/jece/2019/4167890/> ([Online; accessed 12-November-2021])
- Rios-Soria, D. J., Schaeffer, S. E., & Garza-Villarreal, S. E. (2013). *Hand-gesture recognition using computer-vision techniques*. CORE. Retrieved from <https://core.ac.uk/download/pdf/295558123.pdf> ([Online; accessed 4-November-2021])
- Saha, S. (2018). *A comprehensive guide to Convolutional Neural Networks - the eli 5way*. Towards Data Science. Retrieved from <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> ([Online; accessed 10-December-2021])
- Sign language recognition using python and opencv*. (2021). Data Flair. Retrieved from <https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/> ([Online; accessed 15-November-2021])
- Sutoyo, R., Prayoga, B., Fifiia, D., & Suryani, M. (2015). *The implementation of hand detection and recognition to help presentation processes*. Procedia Computer Science. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1877050915020682> ([Online; accessed 4-November-2021])
- Trigueiros, P., Ribeiro, F., & Reis, L. P. (n.d.). *Vision-based Portuguese Sign Language Recognition System*. CORE. Retrieved from <https://core.ac.uk/download/pdf/55638597.pdf>. ([Online; accessed 15-November-2021])

- Wei, J. (2020). *Alexnet: The architecture that challenged CNNs*. Medium. Retrieved from <https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951> ([Online; accessed 5-May-2022])

Appendix A

Source Code of the Application

```
1 all_images=imageDatastore('improvedtestings','IncludeSubfolders',true,'LabelSource','foldernames');
2 all_images=shuffle(all_images);
3 all_images.ReadSize = 100;
4 imdsValidation=imageDatastore('train4','IncludeSubfolders',true,'LabelSource','foldernames');
5 a = transform(all_images,@preproc, IncludeInfo=true);
6 dataPreview = preview(a);
7 montage(dataPreview(:,1))
8 nn=alexnet;
9 layers=nn.Layers;
10 %the first value in layers(23) which is 24 represents the number of letters
11 %in the alphabet
12 layers(23)=fullyConnectedLayer(24,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20);
13 layers(24)=softmaxLayer;
14 layers(25)=classificationLayer;
15
16 opts=trainingOptions('sgdm','InitialLearnRate',1e-3, 'MaxEpochs',10,'MiniBatchSize',100,'Shuffle', ...
17 'every-epoch','ValidationData',imdsValidation,'ValidationFrequency', 3, 'Verbose',false, 'Plots', ...
18 'training-progress', L2Regularization=0.00001);
19 HandTalkerNetv9=trainNetwork(a,layers,opts);
20
21 save ('HandTalkerNetv9.mat', 'HandTalkerNetv9', '-v7.3');
```

Figure A.1: Main Program for training the data set

```

1 function [dataOut,info] = preproc(dataIn,info)
2   dataOut = cell([size(dataIn,1),2]);
3   for idx = 1:size(dataIn,1)
4     temp = dataIn{idx};
5
6
7     tform = randomAffine2d(Rotation=[-5 5],XTranslation=[-50 50],YTranslation=[-50 50]);
8     outputView = affineOutputView(size(temp),tform);
9     temp = imwarp(temp,tform,OutputView=outputView);
10
11
12
13     temp = jitterColorHSV(temp, Brightness=[-0.3 0.3],Contrast=[1.2 1.5],Saturation=[-0.4 0.4]);
14
15     sigma = 1+2*rand;
16     temp = imgaussfilt(temp,sigma);
17
18
19     if rand<= 0.6
20       temp = repmat(rgb2gray(temp),[1 1 3]);
21     end
22
23     temp = imnoise(temp, "gaussian");
24
25     dataOut (idx,:) = {temp,info.Label(idx)};
26
27   end
28
29 end

```

Figure A.2: The preproc function which handles the Data Augmentation

```

1 datastore = imageDatastore('train4','IncludeSubfolders',true,'LabelSource','foldernames');
2 for i=1:length(datastore.Files)
3   input_images{i,1}= imread(datastore.Files{i,1});
4   net_input{i,1}=imresize(input_images{i,1},[227,227]);
5   filename=datastore.Files{i,1};
6   imwrite(net_input{i,1},filename);
7 end
8

```

Figure A.3: Function that resizes all the images into 227 x 227

```

1 classdef StartGame < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5         UIFigure          matlab.ui.Figure
6         WelcomeLabel      matlab.ui.control.Label
7         EnterNameButton   matlab.ui.control.Button
8         TextArea          matlab.ui.control.TextArea
9         Label             matlab.ui.control.Label
10        LifeVal           matlab.ui.control.Label
11        LivesLabel        matlab.ui.control.Label
12        Answer            matlab.ui.control.Label
13        Wordspell         matlab.ui.control.Label
14        Image             matlab.ui.control.Image
15        Letter_Dispatch   matlab.ui.control.Label
16        ExitButton        matlab.ui.control.Button
17        ScoreVal          matlab.ui.control.Label
18        ScoreLabel        matlab.ui.control.Label
19        TimeData           matlab.ui.control.Label
20        TimeLabel         matlab.ui.control.Label
21        StartButton       matlab.ui.control.Button
22    end
23
24
25    properties (Access = private)
26        cam; % Description
27        t;
28        count=1;
29
30    end
31
32    methods (Access = private)
33
34        function toggleButton(app)
35            app.TimeData.Text = num2str(str2double(app.TimeData.Text) - 1);
36

```

Figure A.4: Part 1 of the main code for the actual game framework of HandTalker

```

47     app.t = timer;
48     app.t.Period = 1;
49     load HandTalkerNetv8;
50     app.cam = webcam;
51     while true
52         frame = snapshot(app.cam);
53         snapped=imresize(frame,[227 227]);
54         label=classify(HandTalkerNetv8,snapped);
55         app.Letter_Disp.Text = label;
56         app.Image.ImageSource= frame;
57         wordisp=string(app.Wordspell.Text);
58         wordans=string(app.Answer.Text);
59         if app.TimeData.Text == '0' | wordisp==wordans
60             app.Answer.Text="*****";
61             app.Wordspell.Text="_____";
62             if str2double(app.TimeData.Text)>60
63                 score=3;
64             elseif str2double(app.TimeData.Text)>30
65                 score=2;
66             elseif str2double(app.TimeData.Text)>0
67                 score=1;
68             else
69                 score=0;
70                 app.LifeVal.Text=num2str(str2double(app.LifeVal.Text)-1);
71             end
72             score = score+strlength(wordans)+str2double(app.ScoreVal.Text)
73             app.ScoreVal.Text=num2str(score);
74             app.count=1;
75             stop(app.t);
76             app.TimeData.Text = 'Time Out!!!';
77             if app.LifeVal.Text ~= '0'
78                 app.StartButton.Enable="on";
79             else
80                 highscore = readtable("Book1.xlsx", "Sheet", 1);
81                 highscore=sortrows(highscore,2,"descend");

```

Figure A.5: Part 2 of the main code for the actual game framework of HandTalker

```

81         highscore=sortrows(highscore,2,"descend");
82         lowesthighscore=table2array(highscore(10,2))
83         if str2double(app.ScoreVal.Text)>lowesthighscore
84             message = {'Youve Beaten a HighScore!!!'};
85             uialert(app.UIFigure,message,'Congratulations!!!','Icon','warning');
86             highscore(10,1)=app.TextArea.Value;
87             highscore(10,2)=array2table(str2double(app.ScoreVal.Text));
88             highscore=sortrows(highscore,2,"descend");
89             writetable(highscore,'Book1.xlsx')
90         else
91             message = {'You ran out of Lives!!!'};
92             uialert(app.UIFigure,message,'Game Over!!!','Icon','warning');
93         end
94     end
95 end
96 currentlet= string(extract(app.Wordspell.Text,app.count));
97 if currentlet ==app.Letter_Dis.Text
98     app.Answer.Text=append(app.Answer.Text,currentlet)
99     app.count=app.count+1;
100
101
102     end
103 end
104 end

```

Figure A.6: Part 3 of the main code for the actual game framework of HandTalker

```

106 % Callback function: StartButton, UIFigure
107 function StartButtonPushed(app, event)
108     %timer
109     app.StartButton.Enable="off";
110     stop(app.t);
111     app.Answer.Text="";
112     app.count=1;
113     app.TimeData.Text = '120';
114     app.t.ExecutionMode = "fixedRate";
115     app.t.TasksToExecute = 120;
116     app.t.TimerFcn = @(~,thisEvent) toggleButton(app);
117     start(app.t)
118
119     fid=fopen('dictionary.txt','r');
120     Nrows = numel(textread('dictionary.txt','%1c*[\n]'))
121     linenum = randi(Nrows-1)
122
123     c=textscan(fid,'%s',1,'Delimiter','\n','headerlines',linenum-1);
124     app.Wordspell.Text=c{:};
125
126
127
128
129
130 end
131
132 % Button pushed function: ExitButton
133 function ExitButtonPushed(app, event)
134     closereq();
135 end

```

Figure A.7: Part 4 of the main code for the actual game framework of HandTalker