Goldsmiths
UNIVERSITY OF LONDON

Final Year Project

Development of Hand Gesture Recognition in extreme visual condition

Author: Brahath SHET

Supervisor: Dr Georgios MASTORAKIS

2023

Abstract

Sign Language Recognition plays important role in facilitating human-computer interaction. The development of hand gesture recognition system incorporates advanced computer vision procedures and machine learning algorithms, which function within certain limits in extreme visual conditions. Experiments were conducted to improve prominent stages of achieving accuracy and reliability in poor illumination, extreme contrast, and disorderly background. Using libraries such as Mediapipe and OpenCV would give an impact developing performance. During my experimentation, algorithms SSD and Blaze-Palm with MobileNetV2 from Mediapipe were used to convert BGR to RGB camera detector using OpenCV library accomplishing the accuracies of approximately 85% to 95%. Gaussian noise and reduced brightness were created with my own dataset of eight static sign language hand gestures. Hand gestures used are: stop, fist, OK, thumbs up, thumbs down, peace, rock & call me.

Keywords

Extreme visual condition

Image-processing

Mediapipe Machine Learning Model

Sign-language

Acknowledgements

I would like to thank my supervisor Georgios MASTORAKIS, for his support and guidance throughout the project.

Also, I would like to express my appreciation to my parents for their support and encouragement throughout my academic journey.

Contents

List of Figures

List of Tables

List of Equations

Chapter 1 Introduction

This project develops a sophisticated method of connecting with people of all backgrounds. Sign language is one of them that must be recognised and translated into other languages to remove barriers.

The concept of using machine learning (ML) and hardware components to run software that will learn from humans and demonstrate the outcome of understanding behaviour by interpreting the action into text that will be shown in a selective number of languages. However, there are obligations when the camera must identify the person's hand with soft lighting and poor-quality image processing for the program to run.

1.1 Motivation

I was particularly interested in developing a visual proposal with the goal of developing pattern recognition that will be good for the general population using technology to get used to having simpler ways of solving issues with information. The way this concept was found when I was watching a foreign language broadcast with a sign language interpreter That motivated me to consider evolving a program to recognise hand gestures under adverse circumstances in low quality camera and active illumination controlled in structure light.

1.2 Aim and Objective

The project aims to fill the gap of development and evaluation of advanced tools and use of algorithms that enhance computer vision capabilities in extreme visual environments.

Firstly, using computer vision technology is challenging for maintaining the accuracy with dataset containing rightful answer with different lighting settings. The method to identify a gesture may cause issues in recognizing the movement captured on camera. Therefore, generalised the limitations of an image that would appear when searching through the categories of figures recorded in when it learns on its own and improves for subsequent patterns.

Secondly, the project will focus on image processing objects giving a clear illustration focused on the hand gesture to define their object categories based on the X and Y coordinates when projected on camera. Furthermore, this will be experimented to find the representation of viewpoints and scale the system towards larger archives within the computer vision field.

Finally, critically evaluating the application will be worked recorded or live feeds that are translating the visual data into simpler form through other languages as text giving the message to people in understanding the language.

1.3 Resources requirements

Python is a high-level programming language that is heavily employed in the process of completing this project and lays a significant emphasis on flexibility and simplicity.[1] I have intended to use many libraries that include OpenCV [2], Mediapipe, NumPy [3] and sklearn

[4] along with some frameworks such as PyQt. The use of equipment is going to laptop and camera used for directing the project and camera could be on borrow for temporary use. As per the cost is not certain on how much it may cost because all the equipment is available with me.

I have access to the materials course such as videos on OpenCV [2], deep/machine learning by Adrian NG [5] and books on computer vision and image processing [6] that my supervisor suggested for me in to comprehend the perspectives that will be formed during this project. The dataset sourced from Kaggle, and self-collect data will need to be pre-processed during training to offer a result based on trial walkthroughs to detect and ensure the importance function of the human action model, I must learn when making the projection.

1.4 Methodology

This project I have planned to run an experiment on the processing vision by running different pictures. Along with this will be developing to active learning model and dive into case study by considerate methods of algorithm affine the fundamentals matrix. Also, action research to find the gap by building and give solution for certain testing which needs to be contained for further evaluate of the techniques that show improvements. Additionally, the survey can be used to uncover improvements in gesture datasets that should track people's movements to collect collective data that can be recorded.

1.5 Outcomes and deliverables

Chapter 2: Background Research – This chapter provides information that conducts recognition to fill up the gap of other related projects and depth of exploration into algorithms.

Chapter 3: Specification – This chapter explains the containing list of requirements that is necessary to achieve and explanation of use all the design specification.

Chapter 4: Methodology – This chapter focuses on detailed procedure on implementation of the system program giving the explanation on the visual aspects described in code and characterizes of algorithms to address the problems.

Chapter 5: Implementation - This chapter demonstrates the performances utilised during development creating pilot study and discusses the findings and observations made. The functionality and interface requirements mentioned in the specification are also determined, as well as whether they have been met.

Chapter 6: Conclusion - this delivers the summary of project conclusion of overall accomplished, and failures encountered with a closing statement. In addition, the future direction for this project is enlightened.

Planning the project will involve spending time learning how to create in the upcoming weeks and providing the plan for developing the application that is a minimal viable product for participants who can communicate through hand gestures.

Chapter 2 Background research

In this chapter, research on hand gesture recognition conducts to create new forms of interactive language. Based on the statistics found online around 450,000 people in the UK are deafblind [7].

The form detecting is process and track the perimeter with lights, shades, colours, and textures. Including the function of algorithms to make possibility of images interpretate the solution of using machine learning techniques.

2.1 Image processing

2.1.1 Lighting

The sun is the primary source of illumination outdoors, and since it is so far away, its rays all go parallel to one another in a predetermined direction. It is also simple to use and may be highly effective for both interior and outdoor settings [8]. A surface that faces the source cuts more rays because the rays are parallel to one another. The photos' resolution displays a good deal of detail while also demonstrating how it links to the more "important" global data. This should make it clearer how basic imaging techniques affect how images are interpreted [6].

The brightness of the surface patch in the scene that the image pixel projects will determine the brightness of the pixel in the image. In turn, the amount of incident light that reaches the patch of surface and the percentage of that light that is reflected to governs how bright the patch is recognised by the camera [8].

The image analysis method of the camera responds linearly to illumination levels of average intensity, but exhibits notable nonlinearities for darker and brighter radiance, which mimics the very wide dynamic ranges of real lighting.

In research from Forsyth and Ponce, it is reasonable to assume that the camera reaction is inversely proportional to the surface patch's intensity for the majority of applications.

The terminology of response projecting on the camera the intensity surface patches would be seen while it is written X for a point in space that projects to x in the image, $I_{patch}(X)$ for the intensity of the surface patch at X, and $I_{camera}(x)$ for the camera response at x. where k is some constant to be determined by calibration $I_{camera}(x) = k\ I_{patch}(x)$ [8].

2.1.2 Shades

In pictorial world shades can be inferences variety of properties often requires calibrating the camera radiometrically, so the knowledge of pixel values map to radiometric values. Notice the lighting has turning component while it has few visible shadows. [8] For instance, the hand in the Fig 2.1 is away from the shaded surface shows that the illumination has darker directed components. The left side of the palm's would cover by the rising light as 1 of 1, and only a small region but a small amount of light has lost, identifying <0.5 of 1 [8].

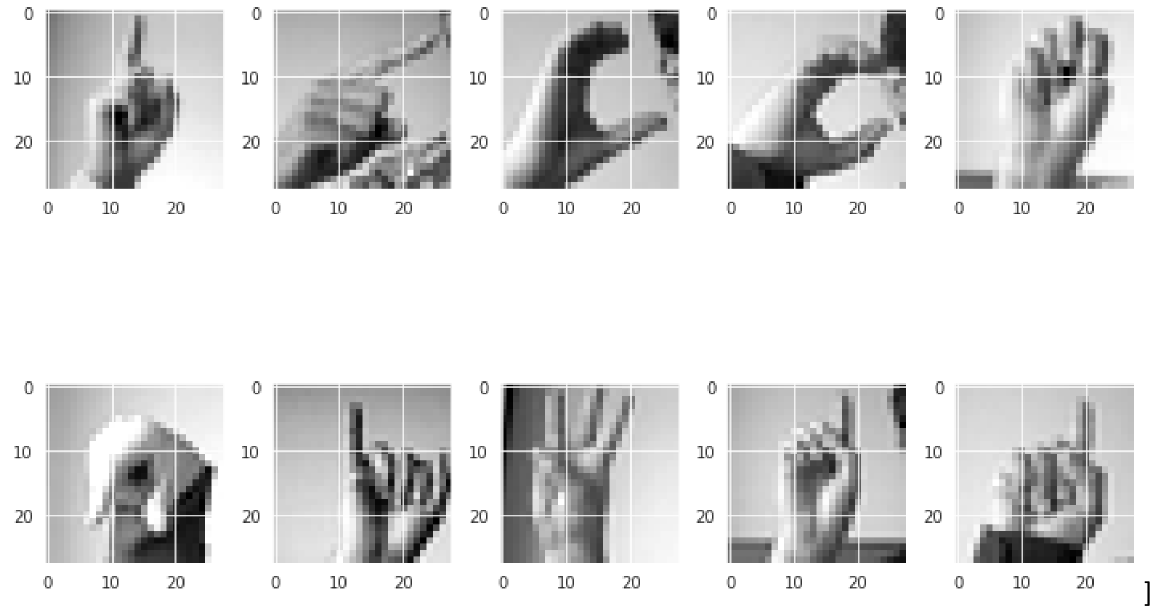

]

Fig 2. 1 static image of alphabets [17]

2.1.3 Pattern recognition

The original input variables are often pre-processed for most applied applications to change them into a new set of variables where it is hoped that the pattern recognition problem will be simpler to solve. For instance, each digit is normally contained within a box of a fixed dimension once the images of the digits have scaled and translated. [9]

The unconstrained imagery we will be thinking about has curved edges and texture at different scales. For instance, hand gestures that involve complex movements, such as signing or finger spelling: These intricate hand positions create curved edges and varied texture that can interfere with accurate gesture recognition. Since, curved lines and textural edges are likely to fragment differently on each image in the series, edges are represented as a collection of straight-line fragments. Because of this, the camera quality will not be able to deliver accurate solutions with descriptions that are of inadequate quality and exclude minor details from the edges. [10]

In Fig2.2 during my research, I came across with these gestures finding it useful to apply into the system for running the experiments.

In Fig2.3 these patterns are shown for camera to record the figures. For illustrations, two up and inverse two up may be the same as characteristic would be based on the abstract shape of the gesture can impersonate with minor changes.

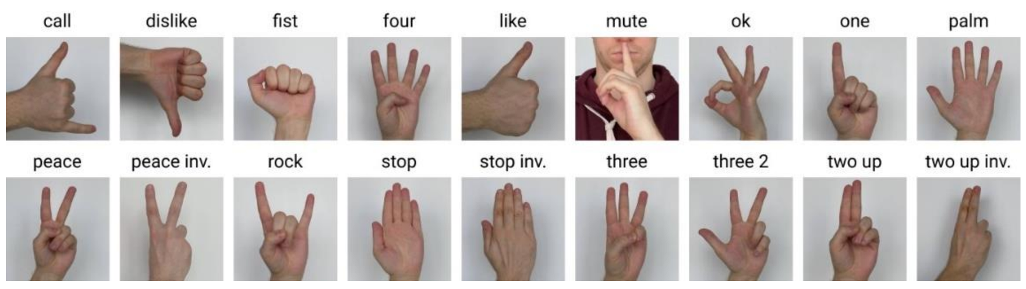


Fig 2. 2 Image dataset of Hand gesture recognition [*11*]

2.2 Languages

According to the British Deaf Association, sign language is not international as there are variation to different languages. Nevertheless, in Great Britain, Ireland, and United States they mostly English as a spoken language [12].

In language, defining a word in language needs to learn and train attentively since they might provide an incorrect response if they have never given it much attention. Dictionaries filled with words, and a good dictionary should have the majority of a language's "words." The sentences that are permitted in a language cannot be found in dictionaries. Because there are numerous sentences in every language, this is not practical. In Fig 2.3, the action of the term “university” at left is BSL indicating as highlights on the design of graduation cap, while offering similarly indications on right in Spanish sign language on concept of pupil clutching books beneath their arm [13].

There are countless motions that needs to be stored giving a thought about surrounding settings with right understanding to translate in terms of texts.

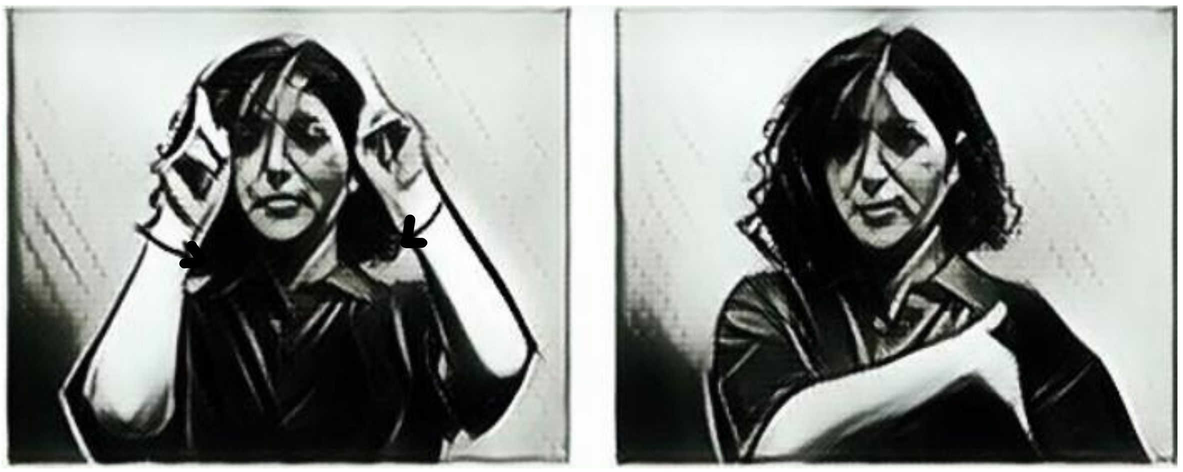


Fig 2. 3 word “UNIVERSITY" (BSL on left) and (Spanish Sign Language on right) [13]

2.3 CNN and SVM

2.3.1 CNN

Convolutional neural networks are used to solve a variety of issues, including those involving image recognition, video analysis, MRI images, and natural language processing. The most powerful deep learning network for image recognition is convolutional neural networks. CNN is effective in extracting image pixels from the collection which can match patterns and identify features like face features and basic shapes such as circles. The CNN architecture is made up of several layers that gather information by compressing input features to generate sets of new characteristics that are shown after navigating nodes inside the hidden layer and providing a sorted output.

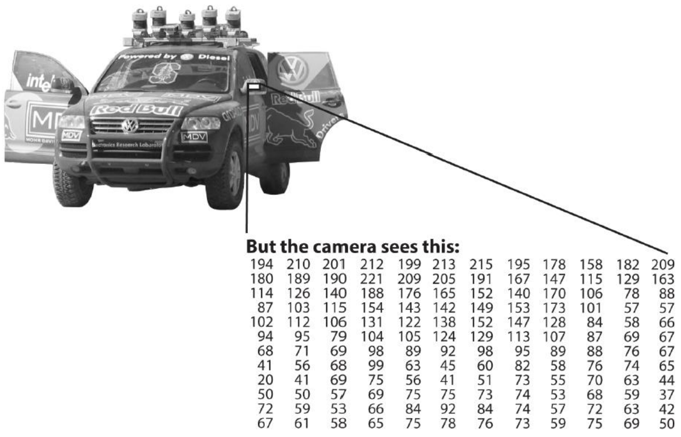
2.3.2 SVM

Support Vector Machines is a machine learning algorithm belonging to supervised learning that analysis data for classification and regression. When the training algorithms develop a model, they classify the input into two distinct groups and attempt to maximise the margin value, which is where they identify the optimal way to separate two layers. [14] Similar studies has come to conclude for hand gestures, employing various algorithms in to help the computer recognise them usually getting higher accuracy when the hand detected. According to research it is shown that SVM overcomes CNN, where it gives best results in classification, the accuracy in PCA- band the SVM linear 97.44%, SVM-RBF 98.84% and the CNN 94.01%, But in the all bands just have accuracy for SVM-linear 96.35% with the dataset of (hyperspectral image) HSI. [15] However, in research from Neil Buckley the system had an 89% recognition accuracy on average when it used CNN algorithms for hand gesture and have an accurate and robust result. [16] Unfortunately, using a dim light or low-resolution camera systems to test and evaluate the potential detection of a hand is an attempt to close the gap because the results can alter depending on how the light is collected from a different angle can vary.

2.4 OpenCV

The primary method of using OpenCV is to run the system that serves as the basis for the architectural design of the programming language to achieve computational efficiency in real-time applications involving different algorithmic areas. This library, which is developed in C and C++ and automatically optimises itself to work on many operating systems, accelerates computer vision and AI with a solid basis in the field. The machine vision system turns video recording data into contextual information representations. [2] For example, observing objects that analyse the image posed the problem of converting the three-dimensional world into two dimensions to reconstruct the signal and obtain a mixture of scenes, because of the variations in brilliance, reflection, and movement, the data would be distorted, necessitating the inclusion of motion sensors in mechanical components to compress the acquired image. [2] Therefore, the work of this library would state purpose of using in practical challenges for this project.

Many approaches for this similar works most of times using position pointers or colours bands this library for having computer vision to run the script written by programmer to get visual recognition of any kind of objects. The project contains of ways of using hand gesture to have limited datasets that will be stored and identify in normal circumstances of particular words.



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Fig 2. 4 To a computer, the car’s side mirror is just a grid of number [2]

Chapter 3 Design Specification

The project aims to develop a machine learning-based system that can recognize and translating hand gestures made in sign language using a camera and software. The system will be designed to work well in difficult conditions, such as low-quality images and poor lighting, to remove barriers for people who use sign language.

Develop and evaluate tools and algorithms that can perform well in extreme visual conditions (such as low-quality images and poor lighting). Use computer vision technology to accurately recognize hand gestures in different lighting conditions. Design and build a camera processing system that can focus on hand gestures and define object categories by their X and Y coordinates. Critically evaluate the application on bases of images or recorded video or live footage and translate the interpreted gestures into text in other languages with the aim of making the message more understandable to people. Use methods such as experimentation, active learning, case studies, and action research to find gaps in current gesture datasets and develop solutions to improve the system's performance.

3.1 Requirements

3.1.1 User requirements

These are the specific needs and expectations that the users of the sign language recognition system have. They may include the ability to recognize hand gestures in various lighting conditions, the ability to translate the recognized gestures into text in multiple languages, and the ability to work with low-quality images.

3.1.2 Requirements specification

This is a document that outlines the specific technical and functional requirements of the sign language recognition system. It may include details on the types of hand gestures that the system should be able to recognize, the languages into which the system should be able to translate the recognized gestures, and the performance expectations for the system in terms of accuracy and speed.

3.1.3 Functional specification

This is a document that describes the specific functions and features of the sign language recognition system. It may include details on the user interface, the algorithms and techniques used to recognize hand gestures and translate them into text, and the hardware and software components of the system.

3.2 Flowchart

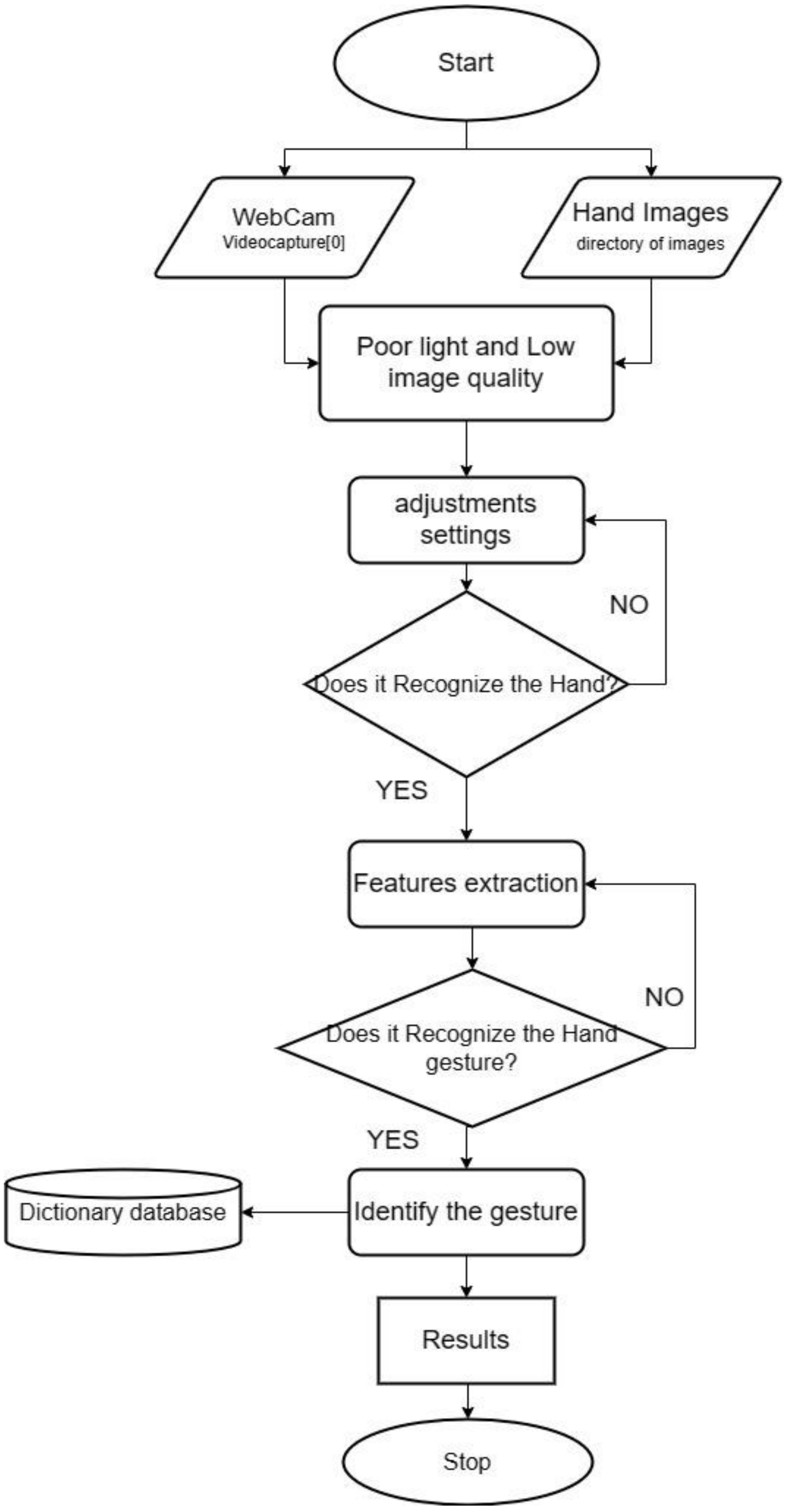


Fig 3. 1 Flowchart of the hand gesture

The study in [18] of the flowchart followed by hand gesture recognition is a way for computers to interpret and understand the meaning of hand movements and gestures made by humans show the steps of a wide range of methods, as I have revised as per my technique to follow when making the flowchart.

The process depicted in this flowchart (Fig3.1) begins by allowing the user to input either a live feed from a web cam or a directory of images. This gives the system the flexibility to recognize hand gestures in real-time or from pre-recorded images, depending on the needs and preferences of the user.

Once the input is received, the system checks for poor light and low image quality, which can significantly impact the accuracy and reliability of the gesture recognition process. If the input exhibits these issues, the system adjusts the light and image settings to improve the quality of the input. This step is important because poor lighting and low image quality can make it difficult or impossible for the system to accurately recognize the hand and the gestures captured through the camera.

After the input has checked and potentially adjusted, the system challenges to recognize the hand in the input. This step involves analysing the visual data and recognising the presence of a hand, as well as its position, shape, and movement. If the hand is recognized, the system moves on to the next step, which is image processing. Image processing involves applying a series of algorithms and techniques to the input data with the intention of extract useful information and features from it. This may include finding specific features of the hand, such as the fingers, palm, and wrist, or analysing the hand's movement and position in relation to other objects in the scene.

If the hand does not recognise, the system adjusts the light and image settings again with the goal of improve the recognition. This step is important because even minor changes in lighting or image quality can affect the system's ability to recognize the hand. By continuously adjusting the settings, the system can improve its chances of accurately recognizing the hand and the gestures.

Assuming the hand recognizes, the system compares the gesture made by the hand to a directory of known gestures stored in a database. This step involves matching the characteristics of the gesture with the stored patterns in the database to categorise a match. If the gesture is in the database, the system displays the result of the recognition. If the gesture is does not exist in the database, the system performs feature extraction, which involves analysing the characteristics of the gesture for the purpose of classifying it and add it to the database for future recognition.

Feature extraction is a key step in the process of building a computer vision system, the particular reason for circumstance that allows the system to learn and adapt to new gestures that it has not seen before. By extracting the key features of a new gesture and adding them to the database, the system can improve its ability to recognize a wider range of gestures over time.

To build a computer vision system like the one described in this flowchart, a developer would need to have a strong understanding of the principles and techniques used in computer vision, as well as the ability to design and implement algorithms and software that can perform the necessary image analysis and pattern recognition tasks. They would also need to be familiar with the hardware and software tools and platforms available for building and deploying computer vision systems, such as cameras, processors, and machine learning frameworks.

Overall, building a computer vision system like this one requires a combination of technical skills, creativity, and problem-solving ability. By following a process like the one outlined in this flowchart, a developer can design and build a system that can recognize and classify hand gestures with a high degree of accuracy and reliability.

Chapter 4 Methodology

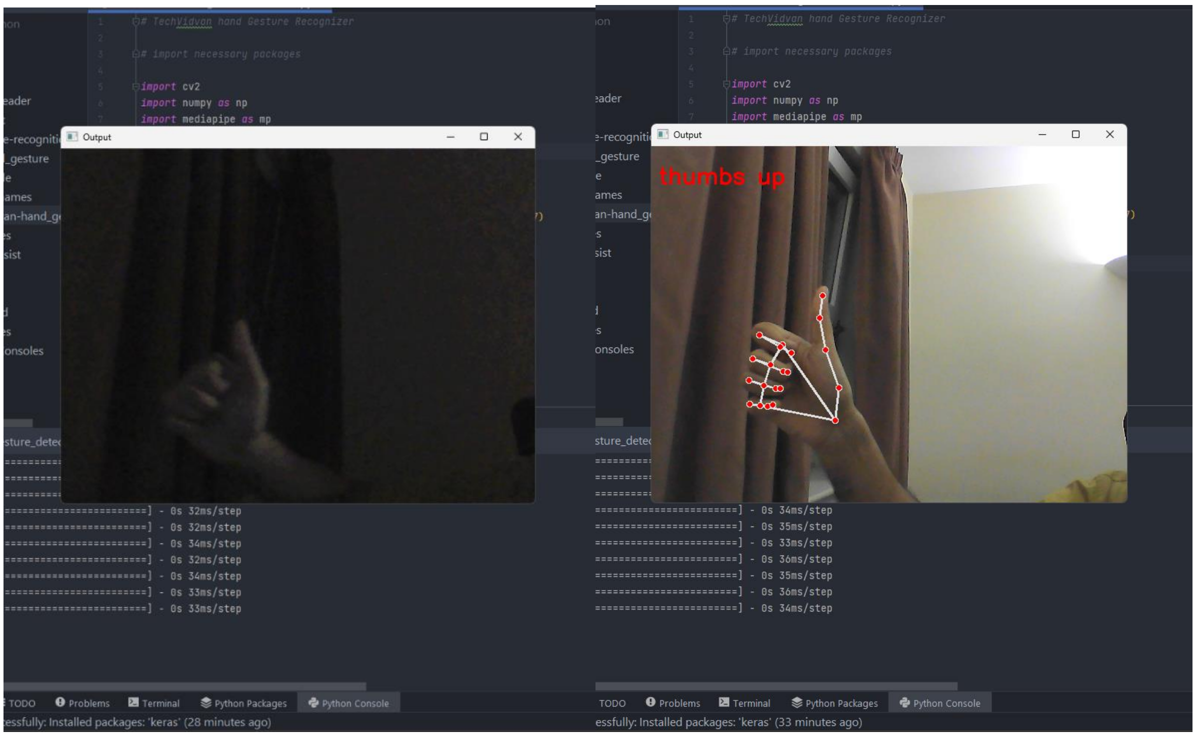
In this section, we will be discussing three algorithms that would be suitable for this system. These algorithms will be pre-trained with a prototype using different architectures to determine which one performs the best in terms of accuracy and efficiency. I will compare the performance of Convolutional Neural Networks (CNN), Support Vector Machines (SVM), and Hidden Markov Models (HMM) on the hand gesture recognition task, evaluating their strengths and limitations to identify the most suitable algorithm for our system and get an understanding in theory. The reason for choosing the study in [19] show using capturing thermal images using Raspberry Pi 4. It achieves high number of accuracies close to 98% using CNN algorithm. However, according to the research from R. Anderson [20] we can see that SVM and HMM show that accuracies differ with HMM getting 87% and SVM 96% in detecting the sign language recognition.

4.1 Analysis of Algorithms

The prototype shown in these screenshots can recognize hand gestures by detecting the hand and using landmarks to find the gesture, as studied in [21]. This is achieved by projecting the vertex of the palm onto the camera.

Mediapipe a framework that build for ML pipeline consist of directed graph of a modular component called calculators which optimised for tasks like model interference and data transformations. However, note that the system requires a significant amount of processing power to accurately identify the outcome. This may be a cautionary factor to consider when using the system to obtain dependable with results. Furthermore, they can run efficiently to run on limited power that have uncertain variation in result, when created new unseen input data and hardware mechanism need to coordinate in steady stream [21].

In hand landmark model can accurately discover the 3D coordinates of the 21 hand-knuckle key points within the detected hand region through regression, which involves direct prediction of coordinates. In the figure 4.1 shown the model can learn a consistent representation of the internal hand pose and is able to manage cases of partially visible hands and self-occlusions [21].



```
# TechVidvan hand Gesture Recognizer

# import necessary packages

import cv2
import numpy as np
import mediapipe as mp
```

```
========================] - 0s 32ms/step
========================] - 0s 32ms/step
========================] - 0s 34ms/step
========================] - 0s 32ms/step
========================] - 0s 34ms/step
========================] - 0s 33ms/step
========================] - 0s 33ms/step
essfully: Installed packages: 'keras' (28 minutes ago)
```

```
========================] - 0s 34ms/step
========================] - 0s 35ms/step
========================] - 0s 33ms/step
========================] - 0s 36ms/step
========================] - 0s 35ms/step
========================] - 0s 36ms/step
========================] - 0s 34ms/step
essfully: Installed packages: 'keras' (33 minutes ago)
```

Fig 4. 1 Making of Prototypes [26]

According to the research from J. P. Sahoo, the convolution process, a filter or kernel is applied to the input gesture image to extract its features. As the filter moves across the entire image, it performs a convolution operation with each pixel of the input image, resulting in the creation of a feature map. The number of feature maps generated is equal to the number of filters used in a convolution layer. This prototype effectively shows that the CNN model developed using five convolution layers, three max pooling layers, and three fully connected layers can accurately classify hand gestures, as shown in Figure 4.2. The deep understanding of the image data demonstrates using this layer architecture in a concept of theory [22].

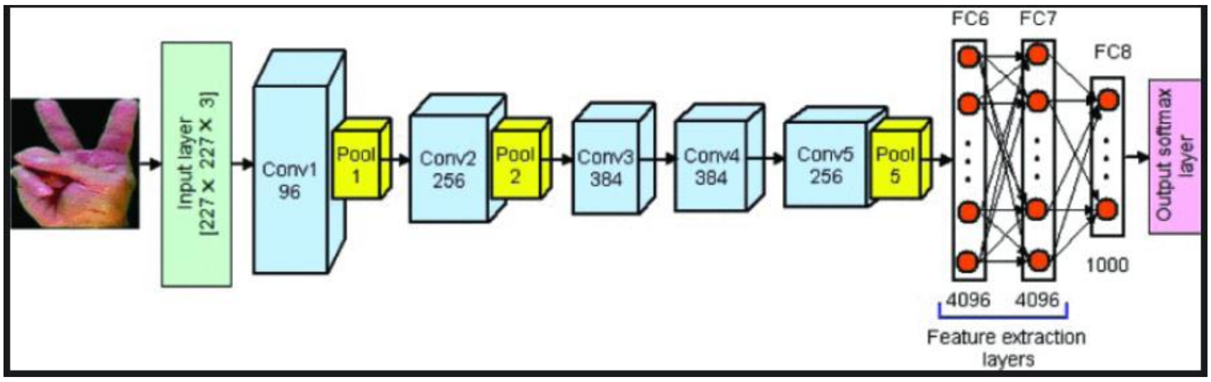


Fig 4. 2 Architecture of pre-trained in CNN [23]

SVMs can also be used for non-linear classification by using kernel functions to transform the input data into a higher-dimensional space. They are often used for image classification tasks due to their ability to manage high-dimensional data and their robustness to overfitting. However, they may not be as effective as CNNs at learning complex patterns in the data. It is also a type of algorithm used for classification tasks. They are based on principles of statistical learning theory and convex optimization, allowing them to classify two classes of elements from a set of training examples. The aim of an SVM is to find a boundary that separates two classes of elements in a data space. SVMs are particularly useful for tasks involving high-dimensional data and can be effective in situations where the data is not linearly separable [23].

HMM, or Hidden Markov Model, can use for tasks involving supervised learning algorithm that uses for pattern recognition tasks. The training process involves presenting sequences of outputs, or training sequences, from a particular system to learn the patterns and behaviours of the system [23].

The recognition of gestures involves the identification and interpretation of natural, continuous movements made by an individual. This process can be challenging due to the variability in the way gestures are performed, as well as the need to segment them in time to accurately understand their meaning. Current systems often rely on a specific starting position in time and/or space to simplify this process, but automatically segmenting gestures remains a perplexing task [24]. Let's say that utilizes a specific starting position is a gesture-based user interface that requires the user to place their hand in a specific location before performing a gesture.

They are particularly useful for tasks involving temporal dependencies, such as hand gesture recognition, as they can model the dynamics of the gestures over time. However, they may not be as effective as CNNs or SVMs at learning complex patterns in the data and may require more data to achieve satisfactory performance.

To sum up everything that has been stated so far, the comparison of these algorithms the use of CNNs will be the most effective choice for tasks involving complex patterns in the data, thus I decided to work on this algorithm. While SVMs may be suitable for tasks with high-dimensional data and HMMs may be the moral choice for tasks involving temporal dependencies.

4.2 Ethics

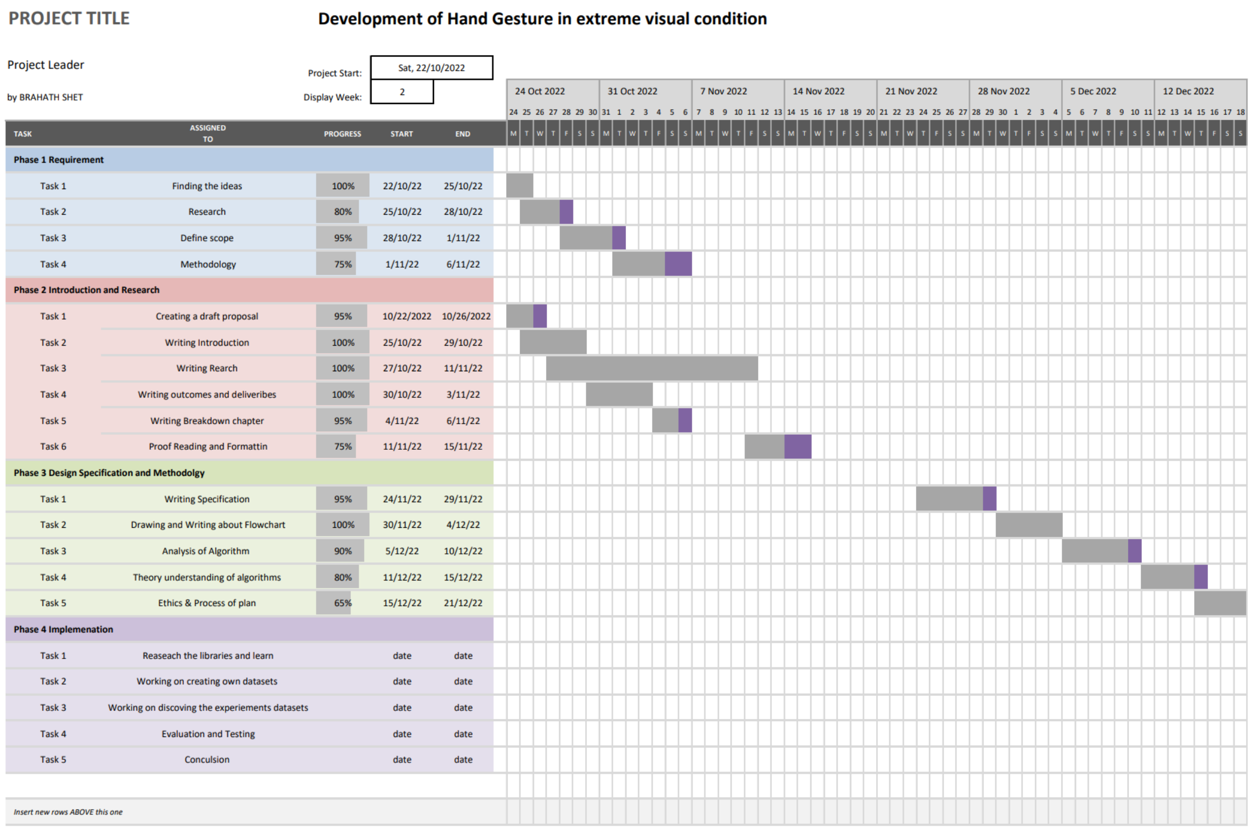
In this project, we will be sourcing the dataset from online platforms such as Kaggle and GitHub. It is important to ensure that the data obtained from reliable sources and that permissions have been obtained for its use in the project. To ensure ethical considerations are met in this project, it is essential to clearly define the aim of the project and ensure it aligns with the needs of the intended audience. It is also crucial to ensure that the work does not violate any privacy laws or regulations, such as the General Data Protection Regulation (GDPR). To do so, it is necessary to oversee personal data responsibly and securely. This includes considering how the data is store and use [25].

4.3 Process and update plan

As part of the development process for this project, I will be sourcing datasets from Kaggle and GitHub to train and assess the recognition system over the time. This will involve gathering and pre-processing the data, as well as implementing various algorithms to improve the accuracy and efficiency of the system. I will be conducting testing and evaluation at various stages throughout the development process to ensure that the system is performing as expected and meets the purposes of the project along with documenting the process by having images that process of results.

Upon completion of training the dataset with algorithms, I will move on to evaluating the system, including any challenges confronted during testing. In this given time, I will be working on the results of gesture also showing the accuracy and loss of recognition.

I present a Gantt chart that shows the planning each process taken to work on completing certain phase in this project. Furthermore, the plan of completing the next phase is target end of March 2023 taking approx. 4-6 weeks to illustration the result and evaluation.



PROJECT TITLE **Development of Hand Gesture in extreme visual condition**

Project Leader

by BRAHATH SHET

Project Start: Sat, 22/10/2022

Display Week: 2

24 Oct 2022 | 31 Oct 2022 | 7 Nov 2022 | 14 Nov 2022 | 21 Nov 2022 | 28 Nov 2022 | 5 Dec 2022 | 12 Dec 2022

TASK	ASSIGNED TO	PROGRESS	START	END
Phase 1 Requirement				
Task 1	Finding the ideas	100%	22/10/22	25/10/22
Task 2	Research	80%	25/10/22	28/10/22
Task 3	Define scope	95%	28/10/22	1/11/22
Task 4	Methodology	75%	1/11/22	6/11/22
Phase 2 Introduction and Research				
Task 1	Creating a draft proposal	95%	10/22/2022	10/26/2022
Task 2	Writing Introduction	100%	25/10/22	29/10/22
Task 3	Writing Rearch	100%	27/10/22	11/11/22
Task 4	Writing outcomes and deliveribes	100%	30/10/22	3/11/22
Task 5	Writing Breakdown chapter	95%	4/11/22	6/11/22
Task 6	Proof Reading and Formattin	75%	11/11/22	15/11/22
Phase 3 Design Specification and Methodolgy				
Task 1	Writing Specification	95%	24/11/22	29/11/22
Task 2	Drawing and Writing about Flowchart	100%	30/11/22	4/12/22
Task 3	Analysis of Algorithm	90%	5/12/22	10/12/22
Task 4	Theory understanding of algorithms	80%	11/12/22	15/12/22
Task 5	Ethics & Process of plan	65%	15/12/22	21/12/22
Phase 4 Implemenation				
Task 1	Reaseach the libraries and learn		date	date
Task 2	Working on creating own datasets		date	date
Task 3	Working on discoving the experiements datasets		date	date
Task 4	Evaluation and Testing		date	date
Task 5	Conculsion		date	date

Insert new rows ABOVE this one

Fig 4. 3 Gantt Chart

Chapter 5 Implementation

5.1 Final implementation

In computer vision and machine learning run the recognition system that can capture and decode image of hand gestures. The project entails live capture of a person's hand motions using a camera, GPU acceleration, and computer vision libraries to analyse the image. Such that it can identify and track the hand's landmarks, using pre-built functions and algorithms in these libraries gives computers the recognising the gestures using machine learning techniques. A system that can distinguish a wide range of task, including gesture recognition, feature representation, and segmentation techniques of hand gestures, including those used in sign languages is to aim in filtering deprived illumination. In this project, I will confer the final implementation of a hand gesture recognition system using Python with free and open-source libraries and GitHub repositories useful to run and experiment the hand gesture system [2][27][28].

During the trial the hand gesture recognition algorithm, a specific number of stages must be achieved. I have produced a datasets of hand movements for the machine learning model to be trained on throughout the data gathering stage. I would extract and store the pixel values from images performing hand gestures using Python tools like OpenCV and Mediapipe which allows us to obtain the x and y pixel positions of the parameter when capturing the data [2][27].

The final implementation of the project involves the following steps:

Data Collection: The first step in building any machine learning project is to collect the data required for training the model. In this project, I have created new data of different hand gestures using a webcam. There are collective data of my hand with different positionings to make the model more robust. Likewise, there are different hand labeled data stored in a format that can be used for training. In terms of inputting the data Mediapipe’ s hand tracking solution does not explicitly use grayscale images as input instead it requires a single RGB camera image which is converted using OpenCV from BGR that contain three colour channel for its hand tracking solution to predict the hand skeleton [27].

Data preprocessing: I used a variety of approaches to improve the data's accuracy at this stage, in addition to noise reduction the data is standardized and made things simpler to examine. Scaling, inversion, and the application of gamma correction are one of the featured techniques utilised to enhance the illumination of the images.

After pre-processing the data, I can move on to the next stage, which is hand detection. This involves using computer vision techniques to identify and track the landmarks of the hand, such as the fingertips, wrist, and palm. There are several computer vision libraries available, such as Mediapipe, which provide pre-trained models for hand landmark detection. I have used these models to extract the hand landmarks from each frame of the video.

The next stage is featuring extraction, where I extract relevant features from the hand landmarks to recognize hand gestures in real-time, the system needs to be able to detect pattern of the hand. Feature extraction is a key step in machine learning, as it helps to reduce the dimensionality of the data and extract meaningful patterns. There are respective convolutional layers that are used to extract features from an input image or video frame. The hand detection model uses a feature extractor based on MobileNetV2, which is a lightweight convolutional neural network architecture designed for mobile and embedded devices [27].

The Mediapipe model uses supervised learning projected ground-truth 3D joints for each hand pose. These 3D joints are projected onto a 2D plane to create synthetic training data. The data combined with personally created datasets, which are generated by logging hand key points and during training the datasets I have dividing the data into 70% training and 30% testing. Therefore, enhancing the model's performance hyperparameters such as batch size, early stopping, activation function. Upon identifying the ideal hyperparameters, the model is trained utilizing an algorithm, for instance, stochastic gradient descent, which aims to reduce the loss function and boost the precision of hand gesture identification.

Evaluating the model: The model must be examined once it has been trained to gauge its effectiveness. But I have given some testing with different noises added to accomplished by putting the model to the experiment on a different visual to classify the break point for which there is no training data. The model's performance can be evaluated using several criteria, including loss and accuracy.

Final implementation: Finally, the hand gesture recognition system is implemented using the trained machine learning model and the hand landmark detection techniques. The system takes input from a webcam, detects the hand landmarks, and classifies the gesture using the trained model. The recognized gesture is then displayed on the screen, allowing the user to interact with the system using hand gestures.

5.2 Detailed technical description.

The Hand Gesture Recognition code is an implementation of hand gesture recognition using a machine learning model. The code is implemented in Python and is designed to run on a machine with a webcam. It uses a pre-trained machine learning model to classify the hand gestures. The code saves the training data in a CSV file that can be used to train a new model.

Technical Description:

The best ways to run these files to use PyCharm IDE with python 3.9 & above and installing libraries.

Libraries: The code uses libraries to perform its functions, including:

argparse: for parsing command-line arguments.

csv: for reading and writing CSV files that have been used for collecting for hand gesture.

copy: for copying objects.

itertools: This Python module offers a selection of tools for interacting with iterators and iterable objects. In addition to functions for working with infinite iterables, it also has functions for filtering and grouping data as well as functions for creating combinations, permutations, and other kinds of iterables [29].

collections: for working with collections of data.

NumPy: Large, multi-dimensional arrays and matrices are supported by NumPy, along with a vast variety of sophisticated mathematical operations that may be performed on these arrays and matrices.

cv2: for processing images and video streams.

Video Capture: The code captures the video feed from the webcam attached to the machine. It is efficient performance allowing the user to set the width and height of the video stream. The video stream is then processed to detect the hand landmarks. In this part OpenCV is a class that provides a convenient way to capture manipulate image and video data efficiently and process video frames from cameras, video files, or image sequences [2].

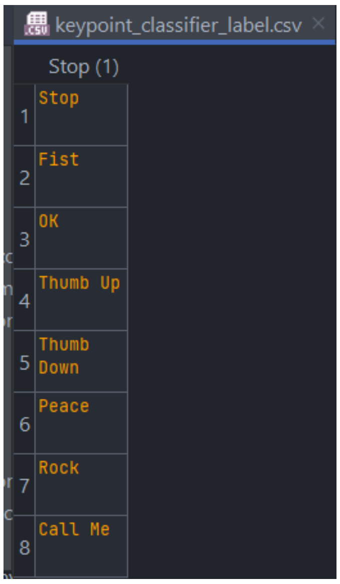
TensorFlow Lite XNNPACK delegate is an optimization feature for TensorFlow Lite, a machine learning framework for deploying lightweight models on mobile and embedded devices. This delegate leverages XNNPACK, a highly optimized library for neural network inference, to improve CPU performance. By using the XNNPACK delegate, TensorFlow Lite can accelerate model execution on devices with limited resources, providing faster and more efficient inferencing [2]. During the process of training and testing the custom datasets converting to TensorFlow Lite will provide a compact model performance with efficient use of resources rather than using TensorFlow GPU power that will performance hungry effective on runtime process [27][28].

Mediapipe: for hand landmark detection and tracking.

According to the Mediapipe, it uses a combination of two models to palm detects employ single shot detector model called BlazePalm and tracks the hand and finger landmarks in an image or video which applies pre-processing steps to prepare input. It can detect 3D landmarks on a hand from just a single frame. The pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box and a hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand key points [27].

Landmark Classification: The KeyPointClassifier models used to classify the landmarks and hand gestures. The class (hand gesture) of the input feature vector is predicted using the classify() method of the KeyPointClassifier. The process accepts a feature vector and outputs the anticipated class label. For example, stop, fist, peace, thumbs up, thumbs down. Based on the identified gestures, the data can then be used to execute specified actions or initiate events in your application while the models are trained using collected dataset, and the code reads the labels for the models from a CSV file [28].

Data Logging: The code logs the hand and finger landmark data to a CSV file for training the models. The logging can be enabled or disabled based on the mode set by the user. The user can also set the number associated with the gesture for logging. The keypoint.csv has all the data starting from 0-7 following the label's structure, for instance if STOP sign would log "0" is associated which marks the coordinates of x and y position values when registering the key point of hand marker. Below is the list of data that are logged into the CSV file.


keypoint_classifier_label.csv

Stop (1)

1	Stop
2	Fist
3	OK
4	Thumb Up
5	Thumb Down
6	Peace
7	Rock
8	Call Me

Fig 5. 1 final list of key points

Accuracy Measurement: The code uses the Counter function from the collections library to measure the precision and stability of detection. it is keeping track of the most recent hand gesture predictions in the past and choosing the most prevalent gesture from that history to serve as the current classification of hand gestures which can be less vulnerable to temporary errors or fluctuations in the input data.

The probability that a hand is present and reasonably aligned in the input image, as indicated by the confidence score. This score is determined by the hand tracker model and depends on various parameters, including the source image's quality and thus the precision of the hand landmark predictions [27].

User Interface: The code has a user interface that displays the video stream with the bounding box and landmarks for the detected hand which is drawn using OpenCV library to create a landmark of the hand when video feed-forwarded to processing it also displays the hand and finger gesture IDs getting the accurate detection.

5.3 Evaluation & Testing

Hand tracking solution is capable of working with both video streams and still images, but the pipeline applied to these different input sources has few differences. The pipeline applies GPU a series of optimizations and methods to improve the performance and accuracy of hand detection when processing a video stream. These optimizations include using temporal information from previous frames to improve hand detection in the current frame.

However, these optimizations may not be as effective when processing a single still image, making it more challenging to detect hands accurately. In the folder "still image test," I have eight different images and test results in "Jupyter notebook" format which show the model may identify the hand. In addition to the code the Mediapipe run the image process iterates over each image converting BGR to RGB, flipping the image coordinates to y-axis for detecting handedness output and label of which hand is selected using the landmark traced.

For example, infrared image of hands a dataset from [30] in file "00_01" Mediapipe does not recognize the hand as the finger are too close and of hand gets bright in the middle so the algorithm may detect as a white circle.

```
Handedness of frame_00_01_0010.png:
None
```

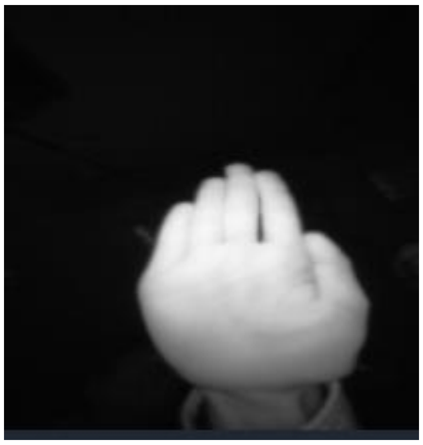


Fig 5. 2 infrared image dataset 00_01 output

On the other part, file "02_02" image of hand from dataset of [30] showing the infrared image gets detected when pointed at the camera as it runs several layers to classify the hand as right with score of accuracy is 0.80 and identify the finger. There are more images that I have provided the results shown in the folder. This is the result output of "02_02" and Fig5.3 shows the pointer of hand.

```
Handedness of frame_02_02_0002.png:
[classification {
  index: 1
  score: 0.8099708557128906
  label: "Right"}]
Hand landmarks of frame_02_02_0002.png:
Index fingertip coordinate: (331.29207611083984, 80.56633472442627)
```

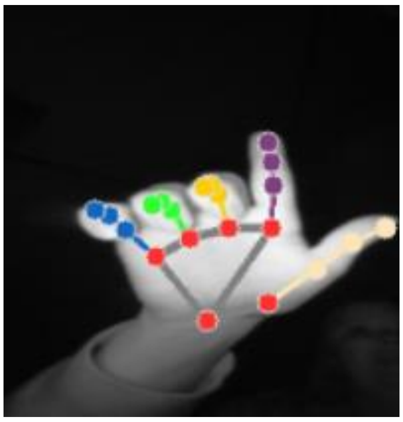


Fig 5. 3 infrared image dataset 02_02 output [31]

For testing, I had taken images of my hand from the files “my_01” and “my_03” to check if the Mediapipe discovers the hand in the normal lighting condition with the hand angled to identify the breaking point. In which in Fig 5.4 “my_01” was predicted and landmark classification as shown in the result below:

```
Handedness of 01.jpg:
[classification {
  index: 1
  score: 0.9954901933670044
  label: "Right"}]
Hand landmarks of 01.jpg:
Index fingertip coordinate: (1180.6285724043846, 835.8670693337917)
```

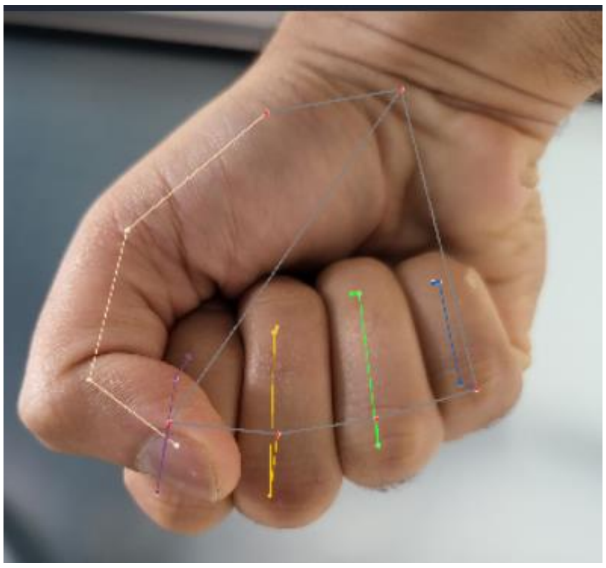


*Fig 5. 4 i*mage output of my_01

However, in Fig5.5 “my_03” with unsatisfactory result by way of the algorithm failed to detect the hand within a static image because with the hand’s symmetrical positioning and angle within the frame. Due to this arrangement, the hand could consider as an object. The image was under default setting image capture from a phone camera [32].

```
Handedness of 03.jpg:
None
```

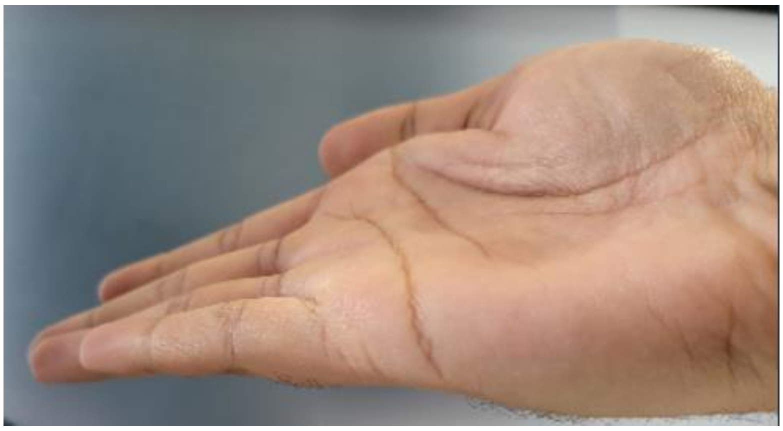


Fig 5. 5 image output of my_03

Therefore, to improve the accuracy of hand detection in still images, adjustments to the pipeline's parameters or preprocessing steps may be necessary. For instance, one can adjust the threshold for hand detection or apply additional image in preprocessing steps, such as contrast enhancement or noise reduction.

Furthermore, performing in real-time I discovered some results that were tested to understand the datasets that were trained in extreme lighting condition showing as proof of how Mediapipe tries to recognize in poor lighting settings of gamma 4.0 and above as it would need more data that can be trained using different visual conditions to make it robust.

In Fig5.6 the arrow marks are showing hand as the system is unable to detect in dark and close range to the camera. Also, in fig5.7 image processing of low light makes it difficult to detect the hand for this model. Given that there are downfalls in succeeding it would need to added max pooling and retrain on hidden layers for the hand to detected. Also, finding out the Mediapipe uses RGB camera as input and dark skin pigment the machine take few moments of time and would need more computational power for a time of extra 2-4seconds to recognize the hand.

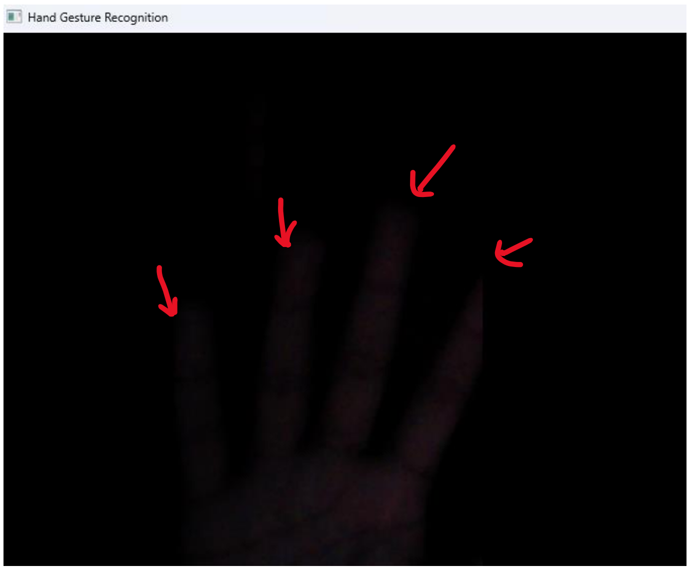


Fig 5. 6 in a dark part the hand undetected.

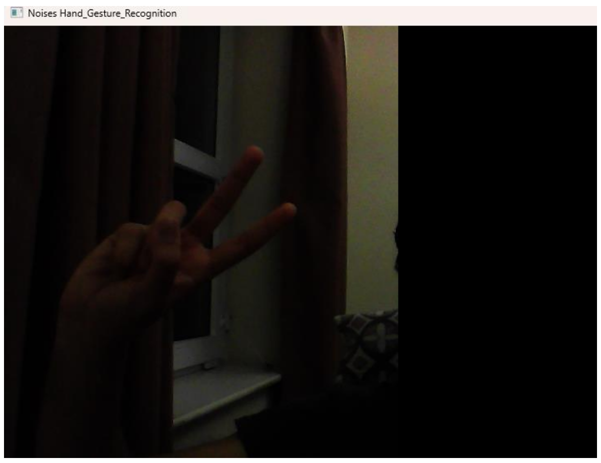


Fig 5. 7 in the 4.0 gamma correction setting the hand is not detected.

During the process of certain datasets giving false labelling seen in fig 5.8 the data collected as it is not perfect data and required data augmentation to regularize the hand in certain position. Given the lighting condition was natural. After data augmentation in fig5.9 the data was calibrated to identify the sign.

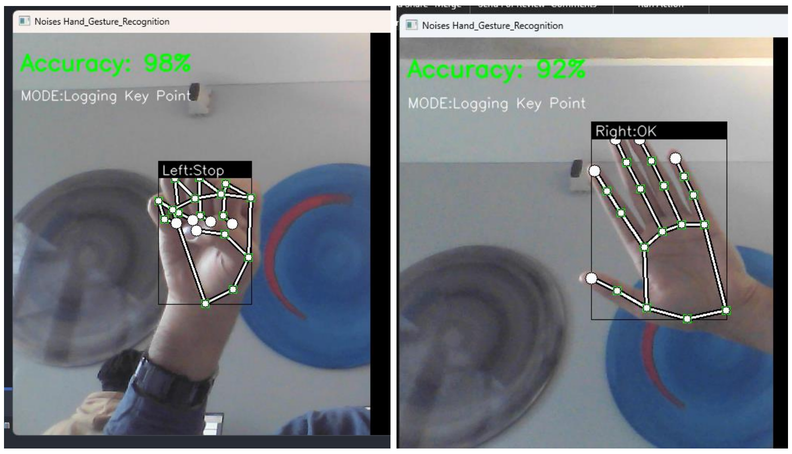


Fig 5. 8 these are false recognition.

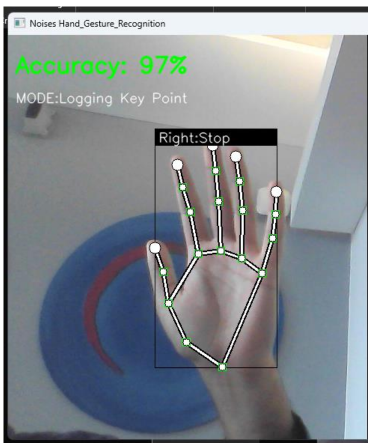


Fig 5. 9 stops sign.

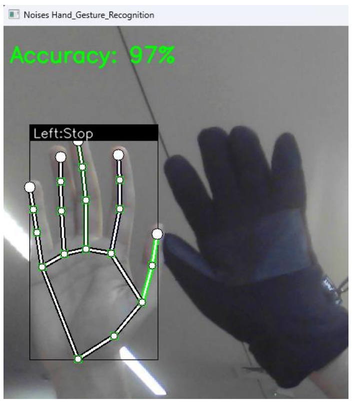
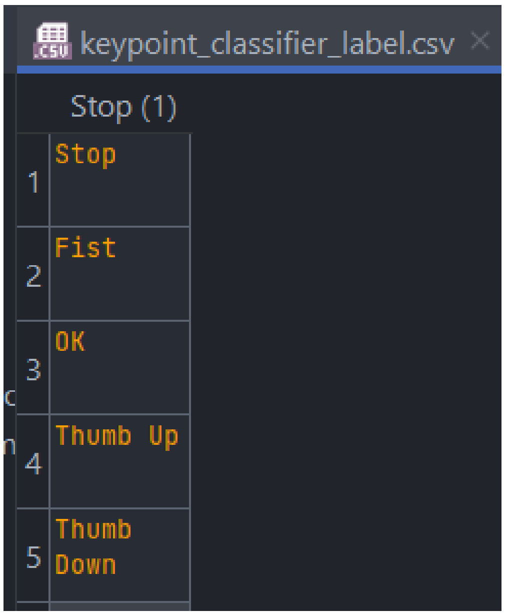


Fig 5. 10 object obstruction for hand

In Fig5.10 the hand detection does not identifies the hand wearing a glove when it has interfered to the model. This would show there is an element of colour that differentiates the hand with the Mediapipe model. The model needs to be trained in hand detecting in both conditions to notice the hands.

While working on the processing and training the data, I ran few tests to check the accuracy of hand detection running the datasets to check how well it will perform. In fig 5.12, during processing new data there was a `loss: 0.4451 - accuracy: 0.7753` given that data is underfitting after 23 epochs, an early stopping added to check these results. The reason to do this test was to get the worst case with five gestures stored into the datasets of fig5.11. This activity demonstrates that the model lacks data, poor data quality, nonrepresentative data, uninformative features, excessively simple models. Hence, more training data or performing data augmentation to increase the size and variety of the dataset can show some sign of improvement.



keypoint_classifier_label.csv

	Stop (1)
1	Stop
2	Fist
3	OK
4	Thumb Up
5	Thumb Down

Fig 5. 11 evaluated on list of labels.

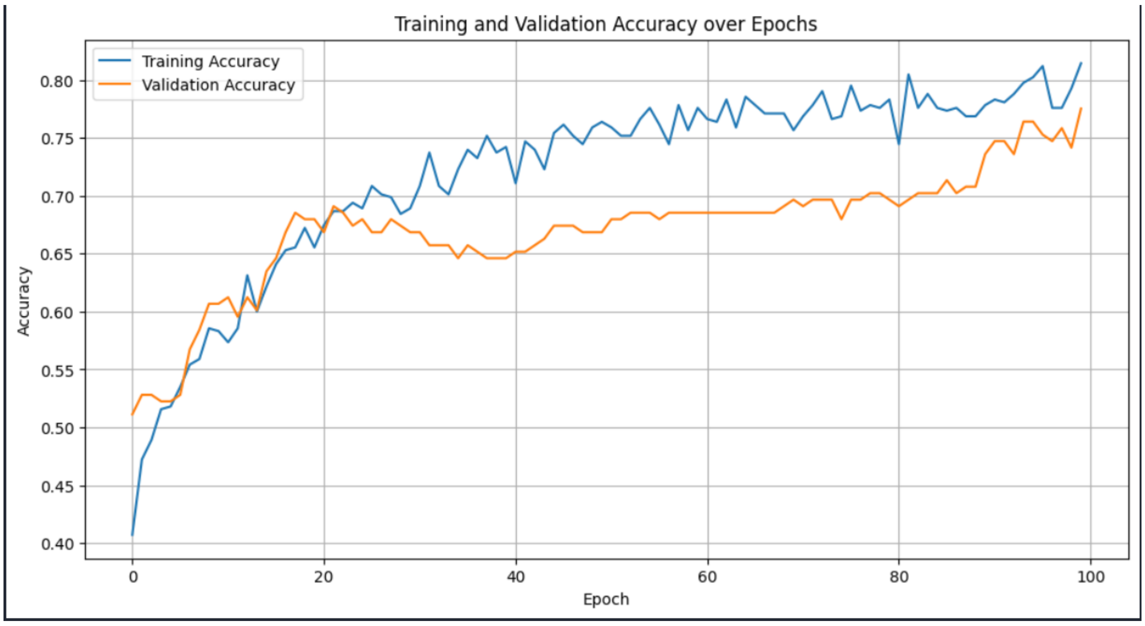


Fig 5. 12 data of the false recognition.

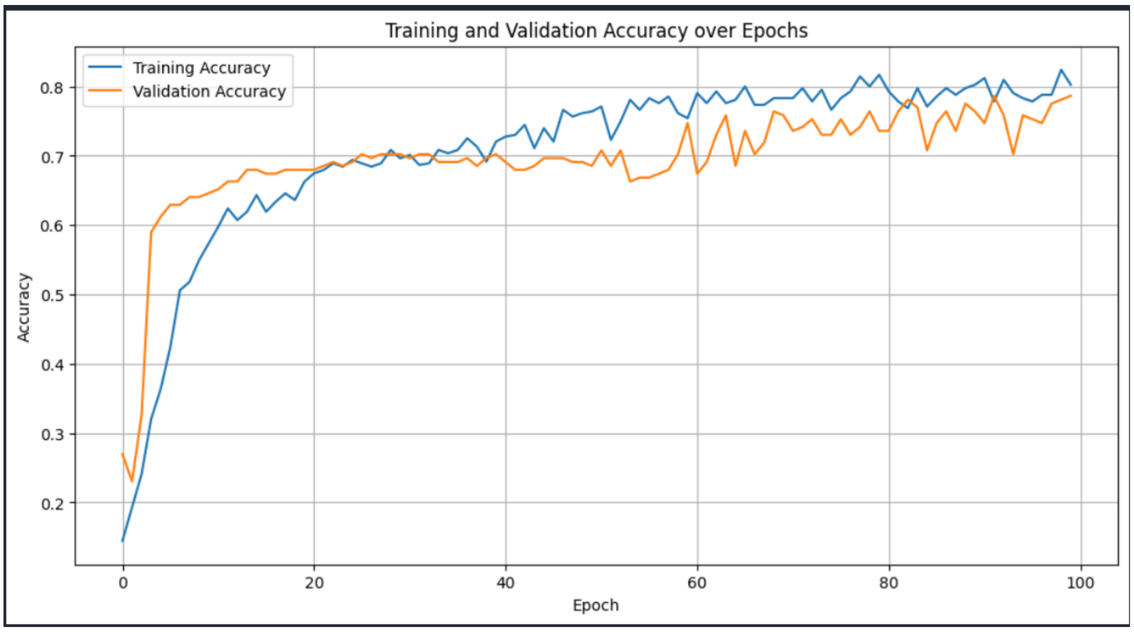


Fig 5. 13 adding more data and use of early stopping on false recognition.

In Fig 5.13, it was retraining on the data augmentation by configuring the stop and ok sign given the `loss: 0.4070 - accuracy: 0.7865` and the data shows it is overfitting at the beginning for the reason that the noise and random fluctuations present in the data, at epochs 0 to 20. However, at epoch 20-30 it gives the good fit showing it perform better than previous fig 5.12. While this data was split into 70% train and 30% test. Therefore, I would focus on tuning the model for further improvement to get better results in the future [31].

In Fig 5.14 this is the final data distributed to all classes presented for analyzing the distribution of classes can help to identify the areas that have imbalance data which classes may require more data. For classes such as class 5 & 6 it has fewer dataset, it might consider collecting more data for those classes to balance the dataset and improve model performance.

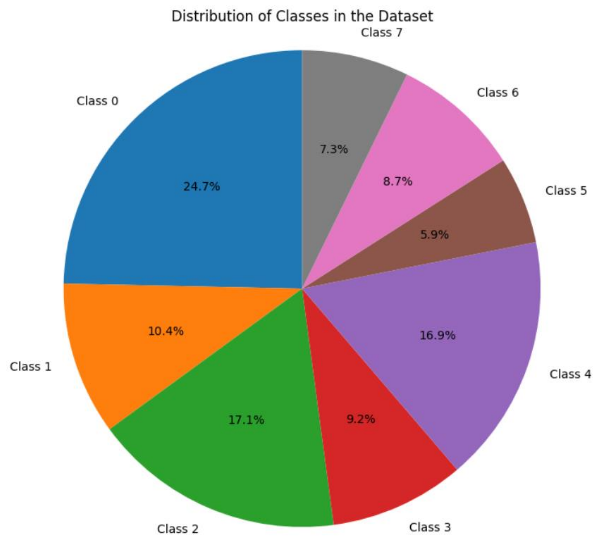


Fig 5. 14 distribution of dataset trained.

To sum up everything that has started so far with implementing of all the points there are improvement needed to make it robust and precision in all types of visual condition would be getting to think over the parts of baseline. Hand Gesture Recognition code is an implementation using a machine learning model. The code intended to run on a machine with a webcam and uses the Mediapipe and OpenCV libraries to capture and process the video stream. The code uses pre-trained models to classify the hand and finger gestures and logs the data to a CSV file for training new models. The user interface displays the video stream and the hand and finger gesture landmark.

Due to the poor image quality and loss of crucial visual information, hand gesture detection in low-resolution situations is a difficult undertaking. Nonetheless, there are several methods and algorithms that may be applied to raise hand gesture recognition's precision in these circumstances.

To increase the precision of gesture recognition, one method is to apply deep learning techniques such as convolutional neural networks (CNNs). Datasets of hand gesture photographs are used to train, which can recognise patterns and features in the data that are crucial for precise categorization. As a result, the network performs better on live feed with low quality and resolution since it can learn to identify patterns even when several information is absent. However, in the still images from Kaggle dataset and my own datasets the Mediapipe library does not identify certain images. It would be less likely to detect the hand in certain angle and light settings when the source of input was changed.

Before submitting the input photos to the recognition system, it is crucial to preprocess them in addition to these methods. The quality of the images can improve and made more suited for precise recognition by using preprocessing techniques including contrast enhancement, noise reduction, and image standardization.

Generally, mix of techniques and algorithms are looked-for obtaining high accuracy in hand gesture detection in low and poor resolution. Even in tricky situations, reliable recognition can accomplish by utilising deep learning techniques, feature extraction, and the right preprocessing.

Chapter 6 Discussion

In the part of the project, it is certain that data will evaluate deeper to comprehend the analysis of the result to the demonstrate the methods that can support and going through further methods of getting the result with different parameters and compare it.

Starting off with the reason of changes made with these settings because to find out if the accuracy and loss with the 80% training and 20% testing datasets and the result are: `loss: 0.5933 - accuracy: 0.8167 - 118ms/epoch - 20ms/step` with higher accuracy shows a better performance model compared to Fig 5.12 and Fig 5.13 giving the predication of hand and the loss is significantly higher than Fig 5.13. Although, the computation time of epoch show it could be complex and may require additional computational resources. For the best trade-off between accuracy and loss this holds the spot for running the experiments.

EPOCHS	ACCURACY	LOSS
20	0.88	0.66
30	0.68	0.7
40	0.8	0.65
50	0.81	0.59

Table 6. 1 showing the varies of data with 80-20

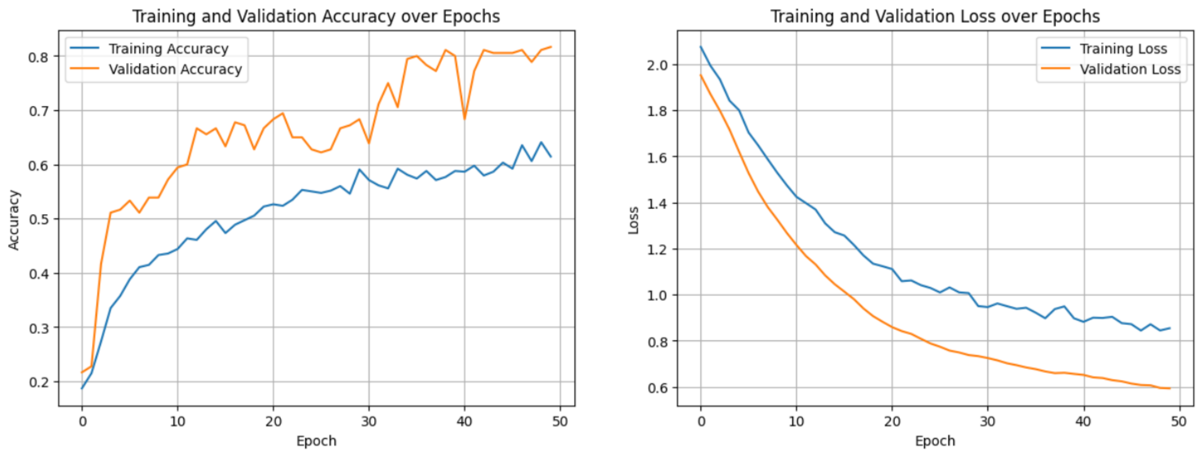


Fig 6. 1 data of Accuracy and Loss 80-20%

Furthermore, one hundred epochs were set with 80% train and 20% test. Because the finding out in changing the hyperparameter shows difference and makes it a good fit for the dataset with these result - `loss: 0.4457 - accuracy: 0.9072 - 142ms/epoch - 20ms/step` showing a better performance when it is runs longer a significant improvement made close to the target. However, the computational resources do not become efficient when 142 per epochs taking more time in training the dataset.

EPOCHS	ACCURACY	LOSS
25	0.63	0.87
50	0.87	0.59
75	0.89	0.50
100	0.90	0.44

Table 6. 2 80-20% with 100 epochs

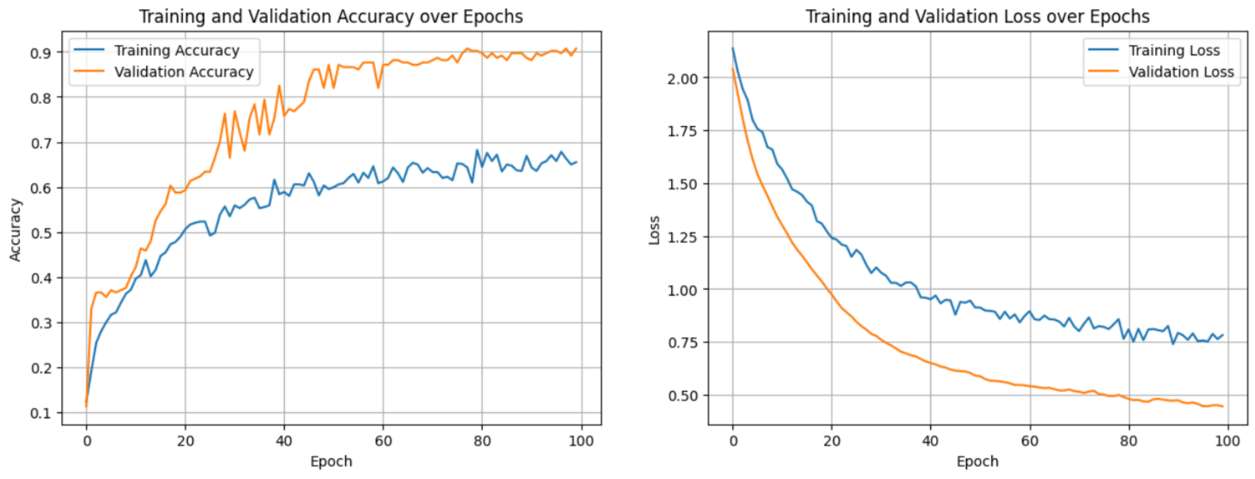


Fig 6. 2 80-20% with 100 epochs

In addition, changes made with the training size with 90% and remaining 10% is testing dataset. The data augmentation was engaged due to the gesture shown in fig 6.4 to find how the data will react showing with enormous difference of `loss: 0.7897 - accuracy: 0.6907 - 43ms/epoch - 4ms/step` this means model's predications is not close to the actual target, there can be inaccurate detection for hand gesture. But with the lower amount of computational resourceful that suggest the model is less complex or less optimized based on training on fifty number of epochs.

EPOCHS	ACCURACY	LOSS
20	0.69	1.03
30	0.67	0.9
40	0.65	0.84
50	0.69	0.7

Table 6. 3 showing the varies of data with 90-10

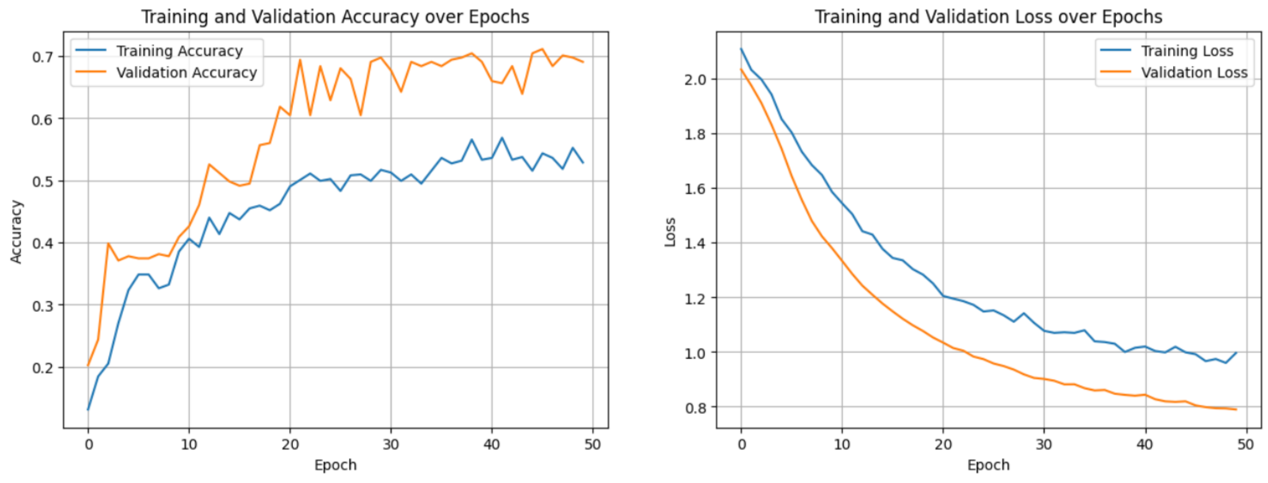


Fig 6. 3 showing the accuracy and loss data with 90-10

Finally, after training and testing the dataset for multiple times and addition new data changes the outcome as showing with accuracy out performance of all and currently holds high above all. This is the `loss: 0.5330 - accuracy: 0.8797 - 32ms/epoch - 3ms/step` for 70% and 30% after working on previous changes. There are reasons to that cause fluctuation in the training loss. The neural nets are trained differently with reduced number of batch size means trusting small portion of data points. If changes are made to update and samples the training may become slow and would need to be checked with the parameters settings to get low losses [31].

EPOCHS	ACCURACY	LOSS
25	0.56	0.9
50	0.79	0.74
75	0.86	0.60
100	0.87	0.53

Table 6. 4 showing the data 70-30

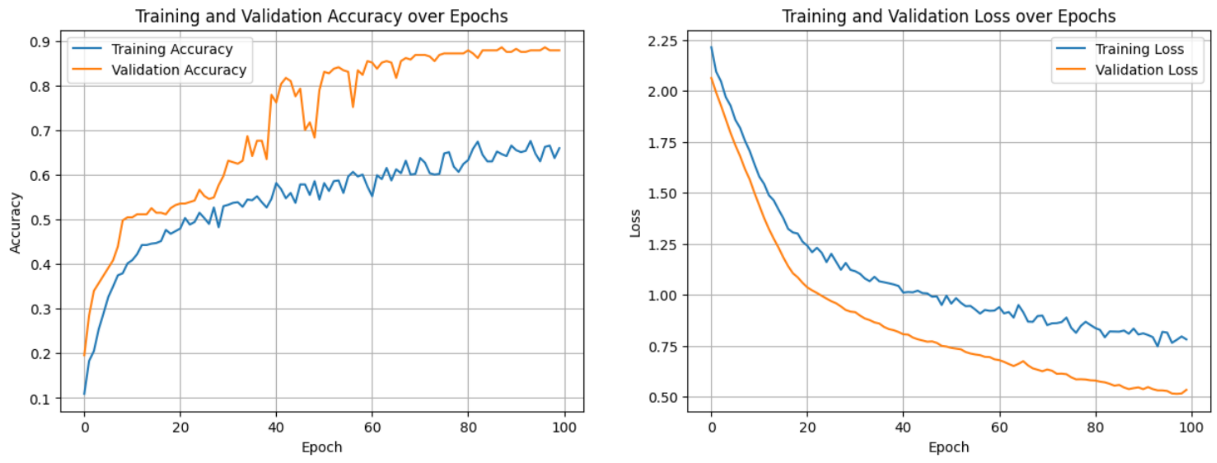


Fig 6. 4 Graph of data 70-30% 100epochs

In terms of vision discussing about the image gives an understanding of how the data would be affect when noises are added such as gamma reduce the illumination and gaussian noise that disturbs the image projection.

This model that is trained to detect the gesture using libraries, it starts off by looking at the data Fig5.14 class 0 (stop sign) based on most amount of data, Mediapipe would recognize the system. In Fig6.5 this shows the gesture predicts palm and rest of the structure of hand giving the accuracy of 71%.

Next, the hand is moved to a corner and follow the detecting to get the breaking point of hand, while looking at accuracy of the positive predication. The Equation 6.1 uses precision and recall of the hand running the equation to ensure the correct gesture that would be useful for classifying hand to true positive and false positive for precision of hand [31].

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Equation 6. 1 Precision and Recall [31]

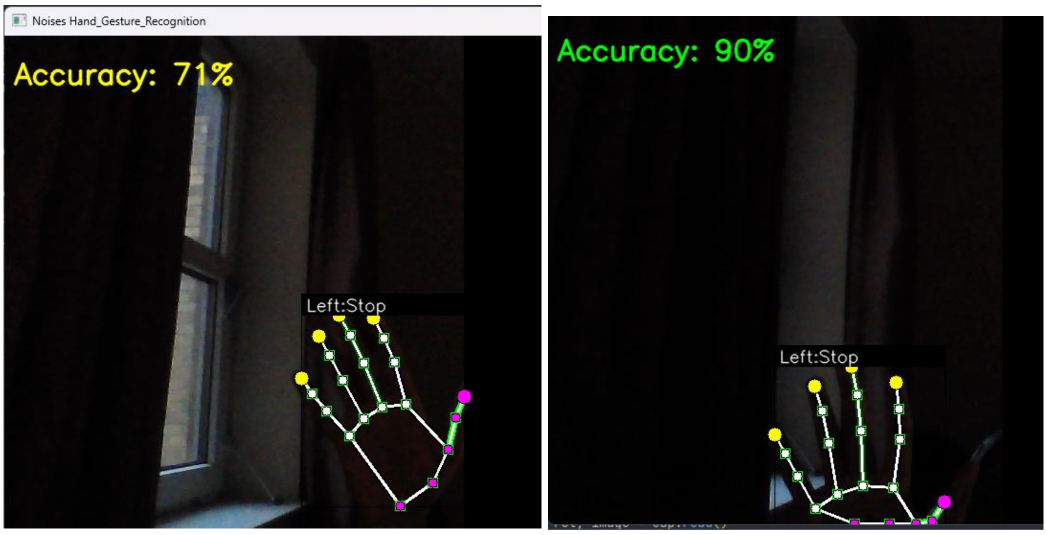
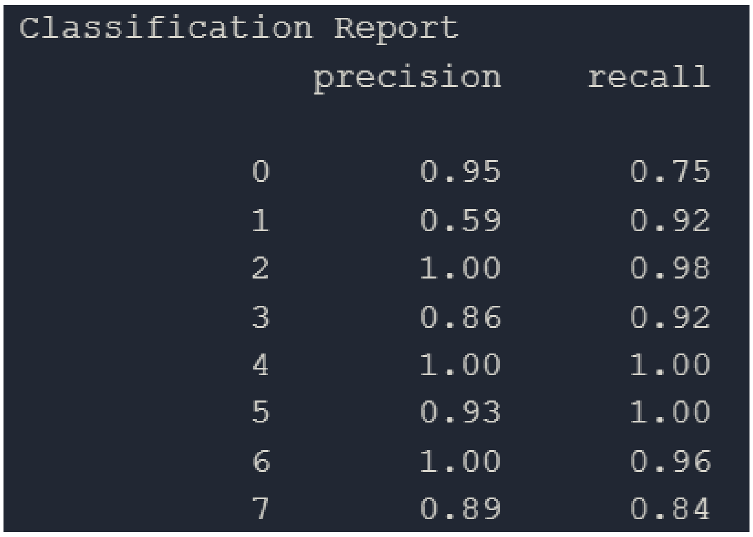


Fig 6. 5 attempting to identify the gesture.

In Fig6.6, this classification report shows that precision and recall for data that was taken after training and testing the model of Table6.4. to show true positive and false positives for each class being represented for the system.



```
Classification Report
              precision    recall

           0       0.95      0.75
           1       0.59      0.92
           2       1.00      0.98
           3       0.86      0.92
           4       1.00      1.00
           5       0.93      1.00
           6       1.00      0.96
           7       0.89      0.84
```

Fig 6. 6 Report of precision and recall for class 8 gestures.

However, in Fig 6.7 hands from a distance cannot be detected makes it difficult for landmark the palm of the hand or the outline of the hand to give a prediction of hand gesture. The settings that were used are gamma >=2.0 and distance of 0.4 meters to 1 meter, meaning that camera was projected to have a black shadow as the output.

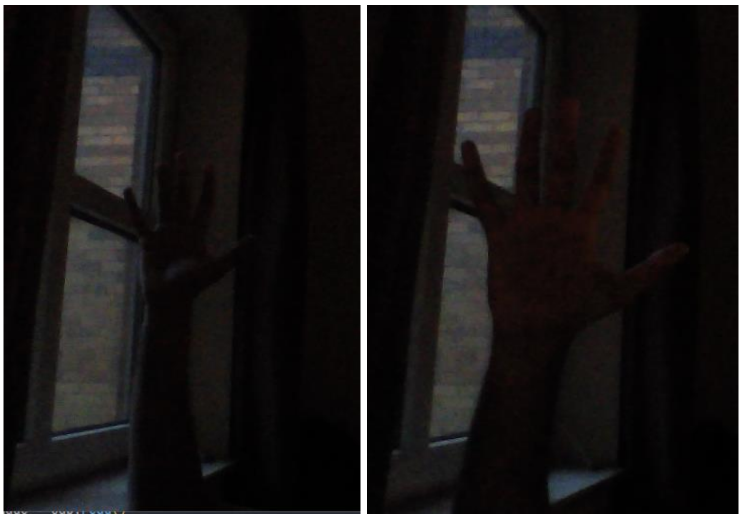


Fig 6. 7 hands cannot recognise at certain distances.

In Fig6.8, finding out if noise misinterprets the hand gesture with gaussian-noise function to image creating a dotted noise simulating a real scenario. The settings used to create noise gamma set to 0.2 and standard deviation 0.45 measuring the amount of variation to generate noise with higher the value of std making it look more distorted. Therefore, the accuracy of hand gesture started with no detection at first, due to a shred of light opening hand detects following close range of distance with accuracy of 89%. The accuracy predicting was higher with the outline of my hand. Other gestures were conducted and finding out the certain gesture were getting either mislabeled to gestures or getting cut-off. To restart the detecting when the palm shows a common diagnose that system picks up with most of the gesture recorded and hand is anticipated on camera taking the RGB primaries.

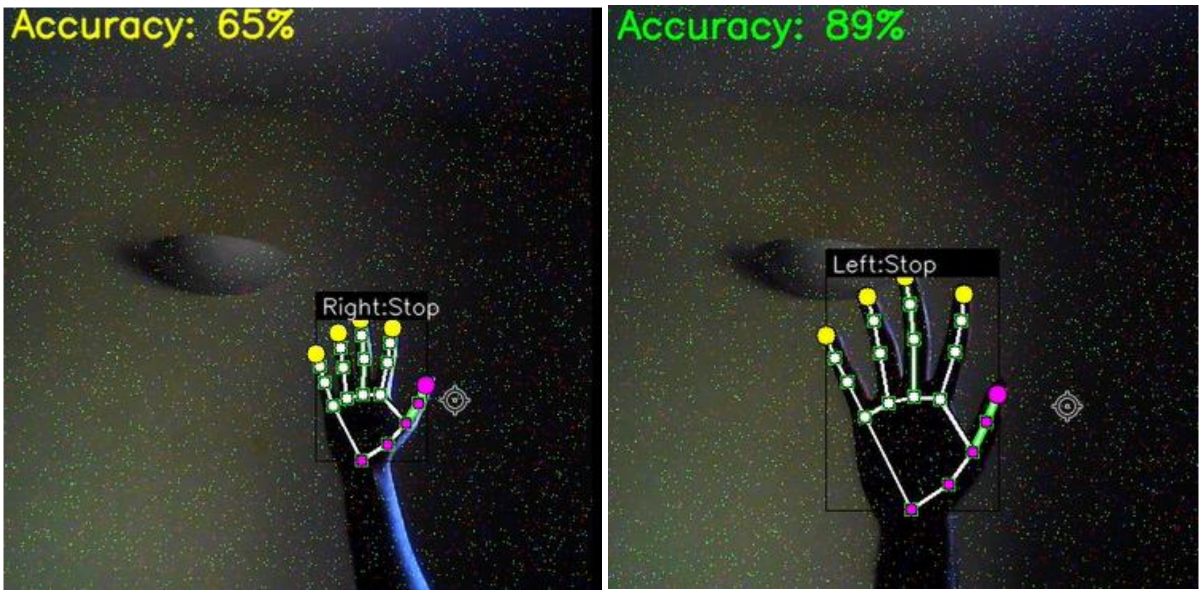


Fig 6. 8 noise added the image gamma 0.2 std=0.45.

When compared to Fig 6.8 the noise is slightly reduced in Fig6.9 which shows that the hand gesture system detects from this slightly reduced noise better. The settings used in this test was gamma=0.3 and std=0.35. Hand was identified in this condition ranging the distance approx. 0.4 to >0.6 meters from the camera but there was stressing parts when the hand was close meaning into a fist in Fig6.10 shows the image processing into a phase that follows with the layers in detecting hand features the first thing noticed with palm after moving on to finger in Mediapipe algorithms.

The reason of adding noise and testing out the robustness of algorithms from Mediapipe and improving on the image processing getting the means of handling gesture system in numerous scenarios. It was an idea to find out by getting a specific point that will distribute after the verge of collapse.

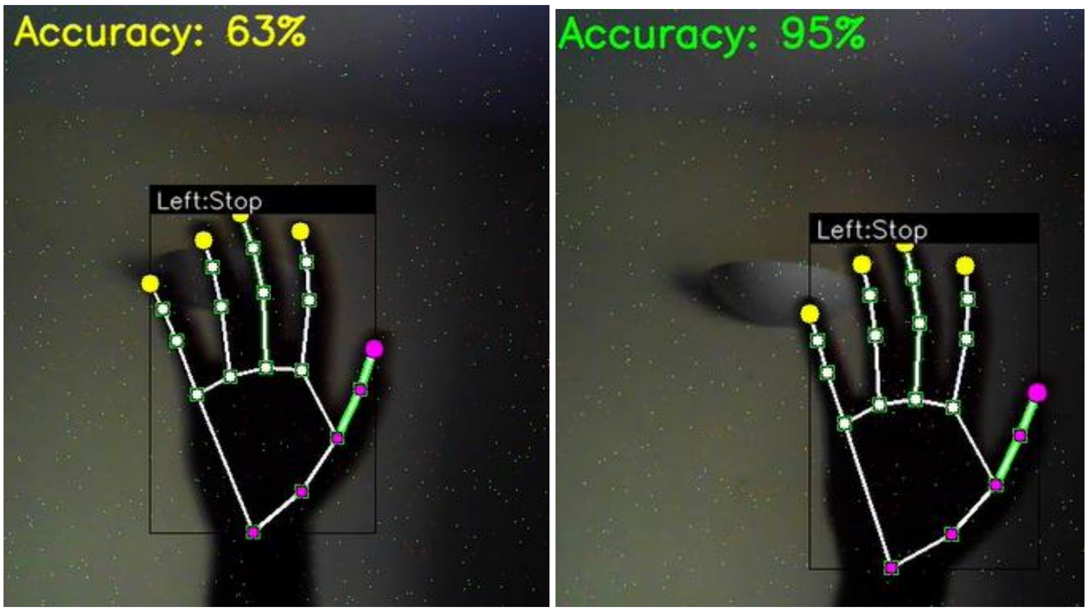


Fig 6. 9 noise added the image gamma 0.3 std=0.35

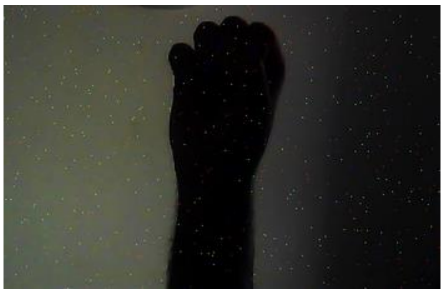


*Fig 6. 10 u*nable to detect with noise gamma 0.3 std 0.35.

Chapter 7 Limitation and Future Works

Despite the promising result achieved in the hand gesture, there remains tasks that is incomplete. The steps that I would establish on developing the project further making the system robust in creating new ways.

Firstly, in this project there was limited amount of time to conduct the experiment that follows a constructive way of perfecting the representation of the gesture system under extreme visual condition. The next logical step in developing would be towards object interference such as wearing gloves that is unable to detect the hand gesture to run different object and tweaking the code into adding landmarks of key point that will classifies the hand layout. It would need be a collection of datasets with diverse types of materials of gloves. The detection would be unnatural with a particular piece, and it records and function good on a certain piece than rest of others for the reason that hands have different shapes and sizes to get all of them will be making a record of all the movements in all types.

Secondly, there are sign language that are advance meaning hand gesture with motion indicating in language while translation. Dictionaries for sign languages with video are available using those libraries could work on advance sign language and adding more data of different signs to identify. Furthermore, developing the concept to translate continuous sentence of all kinds of sign language such as BSL, ASL, ISL etc. and record them into transcripts would work as sign to text, therefore for those people who do not understand the translation could know about the translation that makes it simpler have conversation by learning.

Thirdly, Mediapipe uses the input RGB from the camera, trying out HSV or monochrome converting from RGB modifying the input and finding out the detecting using distinct colour filter such as warm and cool, vintage etc. would likely find out Mediapipe threshold with my datasets created using key pointer. Still, the limitation with Mediapipe was it was not able to detect dark skin pigment with noise to predict the hand gesture. Thus, accumulating more data of diverse categories could improve the system in search of classifying the hand gesture after it is trained and tested on unseen datasets.

Finally, to take further developing in the future an idea for building the system to mobile app will be exciting that can be able to run on portable devices with low efficient power usage and translate sign language on the move.

Chapter 8 Conclusion

In conclusion of this project, the focus was making the use of computer vision libraries and successfully classifying of the hand gesture in real-time vision based capable to translate static sign. During my research, I came across thermal images and infrared that was not cost-effective. Taking the initiative approach in daily life finding the works of making the hand gesture system which would be able to experiment by coding with these noise interference settings. After experiment rounding up to an average of approx. 75%-95% of the accuracy with the use of camera of 720p HD resolution for real-time. The numbers of image and CSV dataset were created according to the lighting conditions that works effectively. Now, the system could be helpful to communicate for deaf and hearing communities to overcome the challenges by breaking the walls of interactions in contrasting scenarios in day-to-day life.

Appendix

There are minor changes made into this project after my supervisor has given feedback on chapter 1 to chapter4 it has being taken to account and immovable to the criteria. In chapter 4 additional details have given to support the analysis of the algorithms that follows with other information. In the evaluation and testing changes have made to give considerate to this project with examples for making the reader know the image and output message that the program constructed by Takahashi.S. The system was developed with my requirement of tools and data for investigating the hand gesture recognition.

GitHub_Repository for this project https://github.com/BrahathS/finalProject.git [32].

Bibliography

1. Publishing, L., & Chan, J. (2014). *Python: Learn Python in One Day and Learn It Well. Python for Beginners with Hands-on Project. (Learn Coding Fast with Hands-On Project Book 1)* (1st ed.). Learn Coding Fast.
2. Kaehler, A., & Bradski, G. (2017). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library* (1st ed.). O'Reilly Media. https://docs.opencv.org/
3. *GitHub - numpy/numpy: The fundamental package for scientific computing with Python.* (n.d.-b). GitHub. Retrieved 12 November 2022, from https://github.com/numpy/numpy
4. *GitHub - scikit-learn/scikit-learn: scikit-learn: machine learning in Python*. (n.d.). GitHub. Retrieved 12 November 2022, from https://github.com/scikit-learn/scikit-learn
5. NG, A. (2022.). *Machine Learning Specialization*. Coursera. Retrieved 10 November 2022, from https://www.coursera.org/specializations/machine-learning-introduction
6. Davies, E. R. (2017). *Computer Vision: Principles, Algorithms, Applications, Learning* (5th ed.). Academic Press.
7. Sense. (2023.). Deafblindness. [online] Available at: https://www.sense.org.uk/information-and-advice/conditions/deafblindness/. Last seen: 29 April 2023,
8. Forsyth, D., & Ponce, J. (2012). *Computer Vision: A Modern Approach* (2nd ed.). Pearson.
9. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Publishing.
10. Harris, C. and Stephens, M., 1988, August. A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, No. 50, pp. 10-5244).
11. Kapitanov, A., Makhlyarchuk, A. and Kvanchiani, K., 2022. HaGRID-HAnd Gesture Recognition Image Dataset. arXiv preprint arXiv:2206.08219. Kaggle. https://www.kaggle.com/datasets/kapitanov/hagrid
12. British Deaf Association. (n.d.). *Help & Resources*. https://bda.org.uk/help-resources/
13. Sutton-Spence, R. and Woll, B., 1999. The linguistics of British Sign Language: an introduction. Cambridge University Press.
14. Huu, P.N. and Phung Ngoc, T., 2021. Hand gesture recognition algorithm using SVM and HOG model for control of robotic system. Journal of Robotics, 2021, pp.1-13.
15. Hasan, H., Shafri, H.Z. and Habshi, M., 2019, November. A comparison between support vector machine (SVM) and convolutional neural network (CNN) models for hyperspectral image classification. In IOP Conference Series: Earth and Environmental Science (Vol. 357, No. 1, p. 012035). IOP Publishing.
16. Buckley, N., Sherrett, L. and Secco, E.L., 2021, July. A CNN sign language recognition system with single & double-handed gestures. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1250-1253). IEEE.
17. CNN using Keras (100% Accuracy). (2020, June 3). Kaggle. https://www.kaggle.com/code/madz2000/cnn-using-keras-100-accuracy
18. Verma, Gyanendra. (2015). Human Computer Interaction using Hand Gesture. Procedia Computer Science. 54. 10.1016/j.procs.2015.06.085.
19. D. S. Breland, A. Dayal, A. Jha, P. K. Yalavarthy, O. J. Pandey and L. R. Cenkeramaddi, "Robust Hand Gestures Recognition Using a Deep CNN and Thermal Images," in IEEE Sensors Journal, vol. 21, no. 23, pp. 26602-26614, 1 Dec.1, 2021, doi: 10.1109/JSEN.2021.3119977.
20. Anderson, R., Wiryana, F., Ariesta, M.C. and Kusuma, G.P., 2017. Sign language recognition application systems for deaf-mute people: a review based on input-process-output. Procedia computer science, 116, pp.441-448

21. Hands Gesture (no date). Available at: https://google.github.io/mediapipe/solutions/hands (Accessed: December 25, 2022).
22. J. P. Sahoo, S. Ari and S. K. Patra, "Hand Gesture Recognition Using PCA Based Deep CNN Reduced Features and SVM Classifier," 2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), 2019, pp. 221-224, doi: 10.1109/iSES47678.2019.00056.
23. M. Y. O. Camada, J. J. F. Cerqueira and A. M. N. Lima, "Stereotyped gesture recognition: An analysis between HMM and SVM," 2017 IEEE International Conference on Innovations in Intelligent SysTems and Applications (INISTA), 2017, pp. 328-333, doi: 10.1109/INISTA.2017.8001180.
24. M. A. Moni and A. B. M. S. Ali, "HMM based hand gesture recognition: A review on techniques and approaches," 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 433-437, doi: 10.1109/ICCSIT.2009.5234536.
25. Queen’s Printer of Acts of Parliament. (n.d.) (2018). Data Protection Act 2018. [online] Legislation.gov.uk. December 23, 2022, Available at: https://www.legislation.gov.uk/ukpga/2018/12/section/2/enacted.
26. Real-time Hand Gesture Recognition using TensorFlow & OpenCV (2021). Available at: https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/ (Accessed: December 26, 2022)
27. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J., and Chang, W.T., 2019. Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172. GitHub., https://developers.google.com/mediapipe
28. Takahashi, S. (2020). hand-gesture-recognition-using-mediapipe [Computer software]. https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe
29. Van Rossum, G. & Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.
30. T. Mantecón, C.R. del Blanco, F. Jaureguizar, N. García, “Hand Gesture Recognition using Infrared Imagery Provided by Leap Motion Controller”, Int. Conf. on Advanced Concepts for Intelligent Vision Systems, ACIVS 2016, Lecce, Italy, pp. 47-57, 24-27 Oct. 2016. (doi: 10.1007/978-3-319-48680-2_5)
31. Géron, A. (2023) Hands-on machine learning with scikit-learn, Keras, and tensorflow: Concepts, tools, and techniques to build Intelligent Systems. Sebastopol (CA): O'Reilly Media.
32. Brahath, S (2023). Final project [Computer vision]. https://github.com/BrahathS/finalProject.git