BSc (Hons) Computer Science
(Artificial Intelligence)

Final Year Dissertation

Sign Language Detection Website

Student:
Krishna Teja Mattapalli
H00313185

Supervisor:
Lynne Baillie

DECLARATION

I, Krishna Teja Mattapalli, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Krishna Teja Mattapalli

Date: 20/04/2022

Abstract

There are different people from different regions and languages living together in this world made hard for people who use sign languages to communicate. In the twenty-first century, it is increasingly crucial to invent new items that make life easier. These days humans are busy and showing more interest in Artificial Intelligence and Machine Learning technologies in all industries for an easy lifestyle. So, it is essential to use Machine Learning technology to help people communicate in sign language. Sign language detection website comes in; research outlines the significant development of a multi-stage technique to detect sign language in real-time. The primary objective is to recognise the signs. This will necessitate machine learning technologies, image processing, and deep learning techniques and techniques to enable visual recognition of the desired elements in real-time. Even though there are several problems associated with this endeavour, comparable solutions have already been explored, laying the groundwork, and providing references for help in this project.

Table of Contents

1 Introduction

1.1 Motivation

Sign language is used to communicate with and transmit messages to members of the blessed community who are deaf or deafeningly. It primarily conveys information through physical communication and bodily movements. A sign language is a collection of organised activities commonly used to communicate among the hearing weekend and hard hearing community. Hand and body component motions are used in sign language gestures (including facial expressions). When compared to conventional languages, each gesture has its meaning. Sign language is a whole language with syntax and grammar [8]. Researchers have previously put forth a lot of effort to recognise sign language. My idea is to develop a website with a sign language detection system.

1.2 Aim

The primary goal of this research is to develop a machine learning model which detects real-time sign language from a website using the system's camera and by manually uploading. The model should be helpful for people who are learning sign language; they can test their knowledge using this model, which helps them learn quickly and people with hearing disabilities have difficulty speaking with other hearing people.

1.3 Objectives

The objectives of this project are:

1. To find the best and most optimal sign language dataset or Collect images and create Dataset by OpenCV through webcam.

2. Find the best CNN model or create a CNN model that gives high accuracy and runs in real-time.

3. Setup TensorFlow object detection pipeline configuration and transfer learning to train a deep learning model to detect through a webcam.

4. Develop a website with integration of the above model.

2 Background

2.1 Sign Language

Humans communicate through natural languages such as speech, writing, body posture, hand motions, head motions, facial gestures, lip movements, etc. Still, Sign language comprises three key components: fingerspelling, word-level sign vocabulary, and body language [11]. Sign language has emerged among speech and hearing persons who have used signs to communicate [8]. For many people, sign language is their primary mode of communication, those who are deaf or hard of hearing. In sign language, movements or gestures are used to express meaning rather than speech. Because of the limits imposed by visual gestures, there are substantial distinctions between signed and spoken languages. Despite this, the two are essentially identical in that they each have their own words and meaning.

2.2 Data Mining and Machine Learning

The study of vast amounts of data to discover patterns and rules is known as data mining [1]. We may provide context to otherwise illegible and complicated data by employing data mining. Data mining may be applied to a seemingly limitless number of applications. Data mining has enhanced several disciplines of study by removing the requirement for human analysis of big datasets and replacing it with machine processing speeds. Data mining and machine learning techniques help to accelerate processes and allow everyone to learn from data that would otherwise be too vast to grasp.

2.3 Convolution Neural Networks

With the emergence of the Artificial Neural Network (ANN), the discipline of machine learning has taken a radical turn in recent years (ANN) [3]. Because the suggested challenge would necessitate real-time detection will need the use of a CNN model that is rapid, precise, and efficient. Most attractive forms of ANN design (CNN), CNNs have advanced at a breakneck pace in recent years. So much so that they outperform typical computer vision recognition systems and, in some instances, outperform humans at specific categorisation issues [2].

2.3.1 Operation

CNN compare pictures by dividing them into little chunks called features. These features are then pixel by pixel compared to a given input picture; the features analyse the image and calculate the resemblance for every component to the quality standard. It is referred to as convolution.

The Rectified Linear Units (ReLU) layer is the most basic layer of a CNN. This step helps to decrease the number of unnecessary computations by simply replacing all negative numbers have a zero: (f(x) = max (0, x)) [4]. It maintains the mathematics clearly and eliminates attempting to compute quantities that approach infinity.

The fully connected layer will be the final layer of a CNN. This layer converts final feature information from a two-dimensional array picture to one set of attributes. The output evaluates every number, including a poll for determining what scores are closest to the result. Results are skewed towards percentages of every deal; instead of the maximum number of wills prevailing, a mean of all the values is compared and calculated for the result before presenting the categorisation outcome. Because the outputs may also be used as inputs, this last linked layer can be layered several times [5][6].

2.3.2 Structure

These three layers are convolution, pooling, and rectified linear units (ReLU) may be layered together in different orders and numerous periods; When a photograph is passed through a convolution layer, it becomes more processed, and when an image is passed via pooling, it becomes smaller. A typical CNN's hidden layer has a mixture of these three layers.

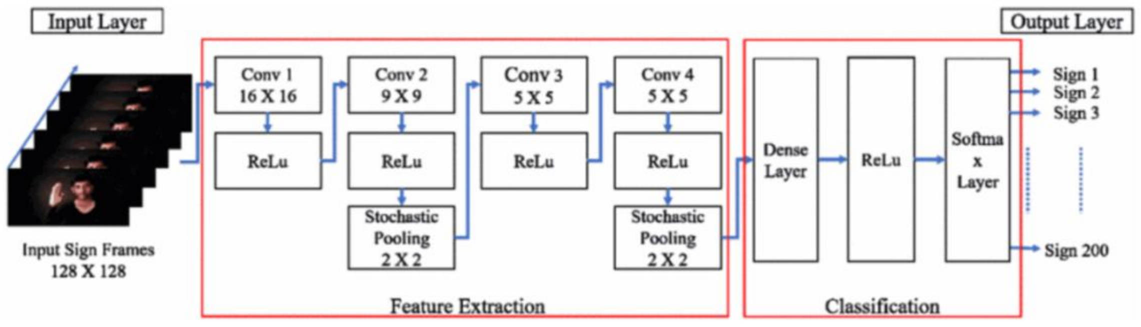


Figure 1- Proposed deep CNN architecture [7]

G. Anantha and his team made up five rectified linear units (ReLu), four convolutional layers, two stochastic pooling layers, one SoftMax output layer, and one dense [7]. Above (Figure-1) shows the adaption of classification that includes ReLU. The system is fed frames with dimensions of 640 x 480. As a preliminary stage, the frames are pre-processed by being resized to 128x128x3 pixels. Resizing input video frames increases the processing capabilities of the high-performance computing (HPC) platform on where the application was running. A 6-node hybrid CPU-GPU processing system was utilised for training the CNN.

Three distinct pooling approaches are performed, including max pooling, stochastic pooling, and mean pooling, and it is discovered that stochastic pooling seems to be the finest for their scenario [7]. With the results of CNN in recognition, the comparison is made with four techniques Mahala Nobis distance classifier (MDC), Adaboost, ANN, and Deep ANN.

2.4 Classifying

Classification algorithms are being used to understand patterns in data discovered by machine learning. Data classification entails assigning labelling to data points, with the labels serving to summarise the data. Clustering is used in different ways by various machine learning approaches. Several categorisation algorithms might be utilised in this project neural networks, decision trees, closest neighbour, and harr cascade.

2.4.1 Neural Network Classification

The Neural Network classification techniques train by identifying features in complex curves in data and imposing a similar decision boundary by looking at designs from the information and drawing general data guidelines. Tags are allocated as needed. The artificial neural network is a bunch of interconnected nodes that can make judgments. The model was developed based on biological examples of neural networks [8].

2.4.2 Decision Tree's Classification

The Decision Tree's Classification approach employs the core notion of a tree structure. The process asks fundamental questions about the data and provides a list of possible replies. After answering the query, the method then descends the tree structure and will request further queries. Consequently, a unique label can be assigned to a data point [9]. As an illustration, if this method was applied to verify your age, a one-layered tree may ask, "How old are you?" and the answer set less than 18, over 18. If the age exceeds 18, the method may label the point above 18.

2.4.3 Closest Neighbour Classification

The Nearest Neighbour classification approach is maybe the most basic of all classification techniques. As it is a supervised learning approach, it necessitates the use of a training dataset. It is indeed a dataset that contains tags, which are often provided by user input. Its collection is much shorter than the one categorised. The concept was straightforward: the dataset's information is combined with the testing set; the difference between every location (point) in both datasets is calculated [10].

2.4.4 Harr Cascade Classification

The function is trained on many negative and positive pictures. As a result, it may be used to identify obstacles in other photos [8], and it can be implemented using the OpenCV python library. This classifier is used for Indian sign language recognition. Prediction is significantly faster than other techniques (support vector machine, hidden Markova model, backpropagation algorithm, and k-nearest neighbour method) with an overall accuracy pace of 92.68 per cent [11]. The system's advantage is that training is performed before actual usage, decreasing processing power and improving effectiveness by analysing quicker throughout testing time. The four classifications from above are significant; it became opted to develop a new categorisation method.

2.5 Clustering

Clustering algorithms are data mining techniques that are used to group items into clusters with comparable values. K-means clustering is a technique that divides a dataset into the k cluster groupings [14].

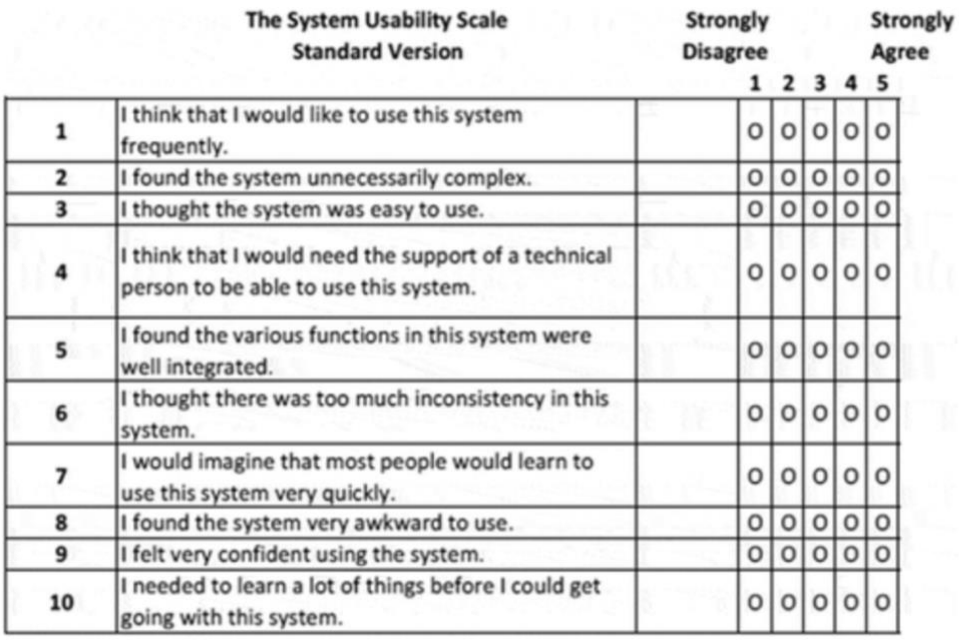
If "k" was four, the methods will accept the dataset and divide it into four clustered categories with Identifiers ranging from 0 to 3. This method straightforwardly divides the data. "K" random data points are selected randomly to be centroids. The remaining data points are then assigned to cluster groups based on the nearest centroid. The clustering quality may be calculated by first computing the difference between several points of data at almost every cluster of collection and evaluating the length among points in the cluster and points not in its cluster. The average difference between pairings in various sets is shared by the average size of couples in the identical group. The more significant the cluster level gives, the better; a cluster value should be more than one [15].

2.6 Django

When creating apps that interact with Web and database servers, web developers frequently employ a Web application framework. Struts (Java), Spring MVC (Java), Rails (Ruby) and Pylons (Java) are some of the most popular frameworks and languages (Python). Despite their relevance in business, however, such frameworks are frequently left out of undergraduate computer science courses [21]. Django, like other Web application frameworks, offers far more Web application support than earlier CGI libraries. It provides logic for features like session management, cookie access, and user logins that are typical in Web applications. Because of its rich features and library support, Django was a breeze to work with. Django's scalability is another significant benefit. It's ideally suited to a large-scale project. Each framework has its own set of limits as well as drawbacks. For smaller apps, Django, for example, is a bit tedious [22].

2.7 System Usability Scale

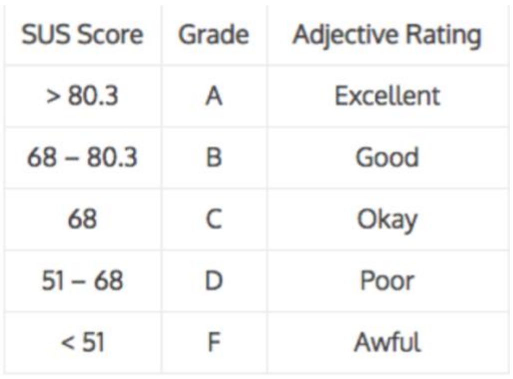
The most extensively used standardised questionnaire for assessing perceived usability is the System Usability Scale (SUS) [23]. SUS offers a number of characteristics that make it an excellent choice for usability professionals in general. The SUS (System Usability Scale) is a frequently used standardised questionnaire for evaluating perceived usability. The SUS was designed to be used in one-off, isolated assessments to produce a single usability and satisfaction score for a product or service. As its usage in the human factors field grew in popularity, other applications emerged and were embraced [24]. The questions used for SUS are below.



The System Usability Scale Standard Version		Strongly Disagree 1	2	3	4	Strongly Agree 5
1	I think that I would like to use this system frequently.	O	O	O	O	O
2	I found the system unnecessarily complex.	O	O	O	O	O
3	I thought the system was easy to use.	O	O	O	O	O
4	I think that I would need the support of a technical person to be able to use this system.	O	O	O	O	O
5	I found the various functions in this system were well integrated.	O	O	O	O	O
6	I thought there was too much inconsistency in this system.	O	O	O	O	O
7	I would imagine that most people would learn to use this system very quickly.	O	O	O	O	O
8	I found the system very awkward to use.	O	O	O	O	O
9	I felt very confident using the system.	O	O	O	O	O
10	I needed to learn a lot of things before I could get going with this system.	O	O	O	O	O

Figure 2- SUS Questionary [23]

SUS score, determine the overall score for all odd-numbered questions, add them all together and remove 5 from the total (X). To get the overall score for all even-numbered questions, add them all up and divide by 25 to get (Y). Multiply the sum of the new values' total scores (X+Y) by 2.5. The score is graded by a score interpreter.



SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

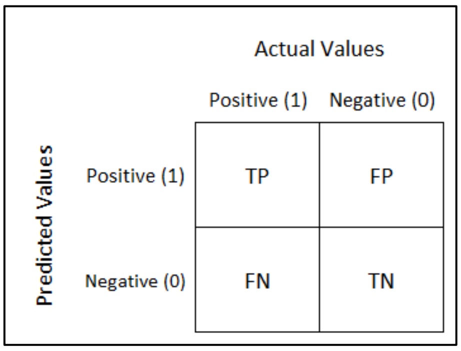
Figure 3- SUS Score interpretation. Source: https://uxplanet.org/how-to-measure-product-usability-with-the-system-usability-scale-sus-score-69f3875b858f

2.9 Related Work

2.9.1 Detection Using Multiple Kernel Learning

A paper by F., Shah. (2021) the proposed technique for recognition of static sign language (fixed pose of hand) of Pakistan sign language (PLS) based on vision-based features; this method uses Multiple kernel learning (MKL) inside of support vector machine (SVM). Different ways of kernel learning are applied in SVM for identification, and the best is selected by performance. It is measured in times of F-score, precision, accuracy, and recall. They developed this system because, in the past, many researchers mainly worked on Chinese, American and Arabic sign language and pointed out how costly use-coloured based hands, sensors and Kinect based approaches.

Researchers built their dataset with thirty-six static letters using sign language videos at 30 frames per second and rotating hand images at different angles, with changes in the hand's position. Researchers got TP, TN, FP, and FN values (Figure-2). Using these values, they measured F-score, precision, accuracy, and recall (Figure-3).



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 4- Confusion Matrix. Source: https://community.alteryx.com/t5/Data-Science/What-is-a-Confusion-Matrix/ba-p/537567

$$Accuracy = \frac{T.N + T.P}{T.P + T.N + F.P + F.N}$$

$$Precision = \frac{T.P}{(T.P + F.P)}$$

$$Recall = \frac{(T.P}{(T.P + F.N)}$$

$$F - score = 2\frac{(Precision * Recall)}{(Precision + Recall)}$$

Figure 5- Formula's [12]

Although it consists of the same dataset employing the images and videos, the image frames captured from the sign language videos are divided in the subsequent step. The split images are converted to grayscale images. Four features are derived from speeded-up robust features, edge-oriented histogram, histogram of oriented gradients and local binary patterns. The idea of multi-kernel learning is developed and classified using a binary classifier named support vector machine to recognise the alphabets, and their performance is analysed. Every peculiarity set is categorised using one of three kernel methods polynomial, linear, and Gaussian and comparisons are made. The primary concept behind the multiple kernels is to find out the best suitable seed and find difficulties. When the multi-kernel method is performed, each kernel learning method is measured. The most suitable kernel learning method for recognising alphabets is chosen as the optimal kernel method for that feature.

2.9.2 Detection Using Computer Vision and Deep Learning

Bantupalli, Kshitij, and Ying Xie (2018) proposed two ways to identify gestures of sign language. The unique American sign language database was utilised for video footage to train the machine to detect motions. The dataset had different movements performed many times. These models recorded video using a convolutional neural network (CNN) to extract spatial features through video stream by applying long short-term memory (lstm) and recurrent neural network (rnn). Video sequence temporal characteristics can be removed using two methods SoftMax and the pool layer of CNN.

The hand motions are recorded using a camera. The movies are then pre-processed to create frame sequences, fed separately to the CNN for two potential outcomes. The global pool layer provides a vector of size 2048, enabling the RNN to examine more properties. The feature sequence is then put into an LSTM, which allows for longer temporal dependencies. RNN struggles from the disappearing or inflating gradient issue, but LSTMs cope with it, leading to greater accuracy on longer data sequences.

These models' problems are with facial characteristics and skin tones, resulting in incorrectly trained features from videos and variations in apparel. As a result, the level of accuracy may suffer. Both are trained to minimise loss when CNN and RNN are assessed separately using a cross-entropy cost function.

3 Requirement Analysis

This section presents some of the fundamental prerequisites for the development of a sign Language detection system. It will focus on functional and non-functional needs and will employ the Moscow methodology to rank the importance of both given requirements. The conditions have been documented, along with their ID and priority. Priorities are divided into the following categories:

Must-Have (M)	Should-Have(S)	Could-Have(C)	Would-Have(W)

Must-Have(M): These are the most crucial requirements to meet while building a system to ensure that the implementation works as planned.

Should-Have(S): These requirements are less critical, but they are still highly vital and take precedence after the initial conditions are completed.

Could-Have(C): These criteria are not strictly necessary for the model to perform appropriately, but they may improve various elements of the implementation.

Would-Have(W): These requirements are not required, but they should be examined if extra time is available.

3.1 Functional

The functional requirements are criteria that a particular implementation must meet to be completely functioning.

ID	Description	Priority
FR1	The website is supposed to recognise various types of signs.	M
FR2	The website should predicate accurately.	M
FR3	The website should take input through a webcam.	M

FR4	The website must be able to allow users to upload images manually from local storage.	M
FR5	The website should recognise signs in dim light.	S
FR6	The website should print the text of the detected signs.	M
FR7	The website should access a webcam.	M
FR8	Responsive User interface.	M
FR9	The website should apply masking on webcam input.	M

3.2 Non-Functional

These requirements are criteria that a specific implementation must meet, generally prescribing how it will act and operate while also giving the system extra capabilities.

ID	Description	Priority
NFR1	The system should predicate 100% accurately.	S
NFR2	The system must produce results quickly.	S
NFR3	The system should work on all web browsers.	M
NFR4	Graphical User interface.	C
NFR5	The system can take information from the directory.	W
NFR6	The design should be easy to use.	S

4 Project Management

This chapter focuses on how the project will be managed.

4.1 Timetable

This section lays out a schedule for how the project's work will be distributed across several months.

Team Gantt [20] was used to construct all the Gantt charts. The project is divided into five stages.

Note: Full Gannt Chart is available in the Appendix.

4.1.1 Data Set Creation

The goal of this phase is to locate or create a new data set. As the sign datasets are organised, this will also set up future investigations. This should be finished by 21/01/22.

4.1.2 CNN Implementation

This phase is dedicated to the creation of CNN and the deployment of the trained CNN. As the sign datasets are organised, this will also set up future investigations. Furthermore, a basic 'template' will be built that will be followed when successive CNNs are developed. On a base upload photo and a camera, I tested real-time performance. This should be finished by 18/02/22.

4.1.3 Detection Model

This phase focuses on the construction of a sign language detection model for both upload and webcam options utilising the trained CNN model from the previous stage.

4.1.4 Website Development

This phase focuses on the web site's construction and the integration of the detection models developed in the previous stage.

4.1.5 Experimental Phase

This phase incorporates the previous phases' creation and research and focuses on implementing the assessment technique mentioned in (Evaluation Strategy). This should be completed by 01/04/22.

4.1.6 Write Up

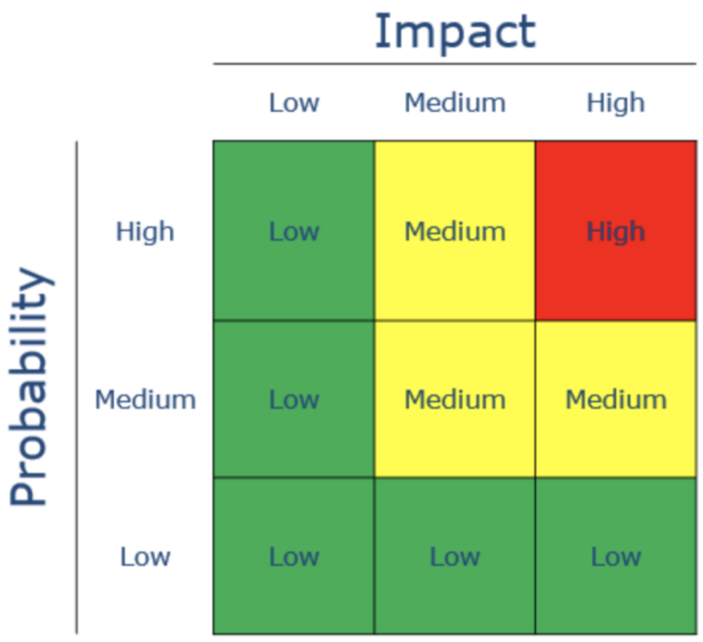
Throughout the project's growth, the dissertation report will be regularly updated. This would assist in reducing load as the deadline approaches.

4.2 Risk Analysis

This section aims to discover and become acquainted with various risk scenarios that may affect the work. The mission's timeline, efficiency, supplies, and effectiveness might all be impacted. If the hazards are detected early, the team can take steps to mitigate the risk's impact.

Throughout the risk analysis, choices must be made about the possibility of a risk occurring and its effect on the model. The categories low, medium, and high are used to describe the impact and likelihood. The intensity of any danger is determined by its impact and likelihood.

4.2.1 Risk Matrix



Probability \ Impact	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

Figure 6 -Risk Matrix. source: https://www.stakeholdermap.com/risk/risk-assessment-matrix-simple-3x3.html

4.2.2 Risk Scheme

- Minimisation tactics try to lessen the risk's impact.
- An escape strategy aims to lessen the likelihood of the risk occurring.
- Acceptance and mitigation of retention
- The planning process entails preparing for the worst and keeping a plan in place to cope with it.

4.2.3 Risk Register

ID	Description	Impact	Probability	Type	Risk
R1	Poor real-time efficiency, inferior accuracy	L	L	Technical	L
R2	Missing deadlines	M	M	General	M
R3	Unexpected university closures	M	M	General	M
R4	There isn't enough data to attain excellent accuracy	H	M	Technical	M
R5	The system is incapable of detecting indications of sign language	H	L	Technical	L
R6	Code files getting lost or corrupt	H	L	Technical	L

4.2.4 Mitigation Strategy

ID	Description
1	Make sure to get good results in the evaluation strategy stage
2	Try comparing work against the initial plan regularly, and gradually increase assigned time per week to the work to try and catch up.
3	Try not to rely solely on lab computers; instead, keep backups on home PCs. In addition, I must verify that I have the necessary programmers if I cannot use the lab computers.
4	More investigation on the sign language dataset will be needed.
5	Ensure everything is working in each stage before going to the next step.
6	Back up code regularly on GitHub.

5 SYSTEM DESIGN

It is a conceptual model that specifies a system's structure, behaviour, and other aspects. A formal description and representation of a system arranged in a way that facilitates reasoning about the system's facilities and behaviours are known as an architectural description. The images below depict how the system is constructed and how it works.

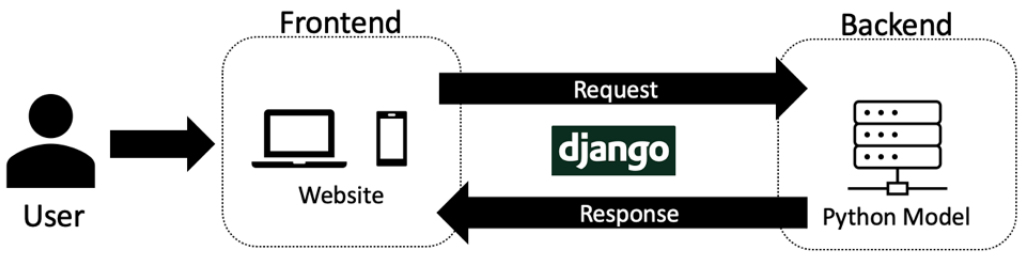


Figure 7 – System Architecture_1

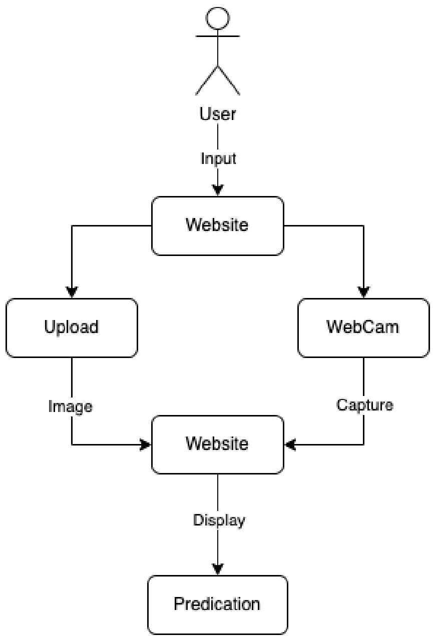


Figure 8 – System Architecture_2

5.1 Data Flow Diagrams:

A data-flow diagram is a graphical depiction of data as it flows through a system or process. The DFD also provides information on the inputs and outputs of each entity, as well as the process itself. A DFD model uses a highly predetermined set of primitive graphics to speak to the capabilities performed by a framework and the information flow among the capabilities. The main reason for the DFD method's popularity is likely because it is a fundamental formalism that is simple to understand and use. DFDs may also be used to understand how information is handled.

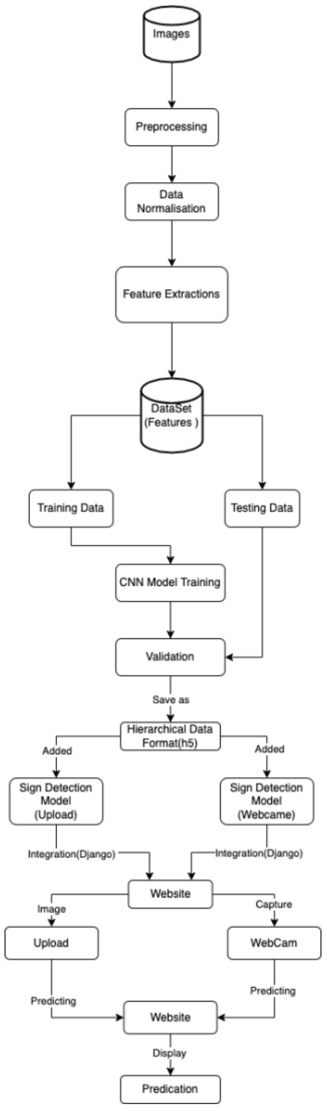


Figure 9 – Data Flow Diagram

5.2 Sequence Diagram:

A sequence diagram is a visual representation of a series of events. Modelling Language is a communication diagram that depicts how processes interact with one another and in what order. Sequence diagrams are also known as event follow diagrams, event circumstances, and timing diagrams. To codify the framework's behaviour and visualise the relationship between articles, sequence diagrams are used. They are helpful in identifying additional questions that arise throughout the utilisation situations.

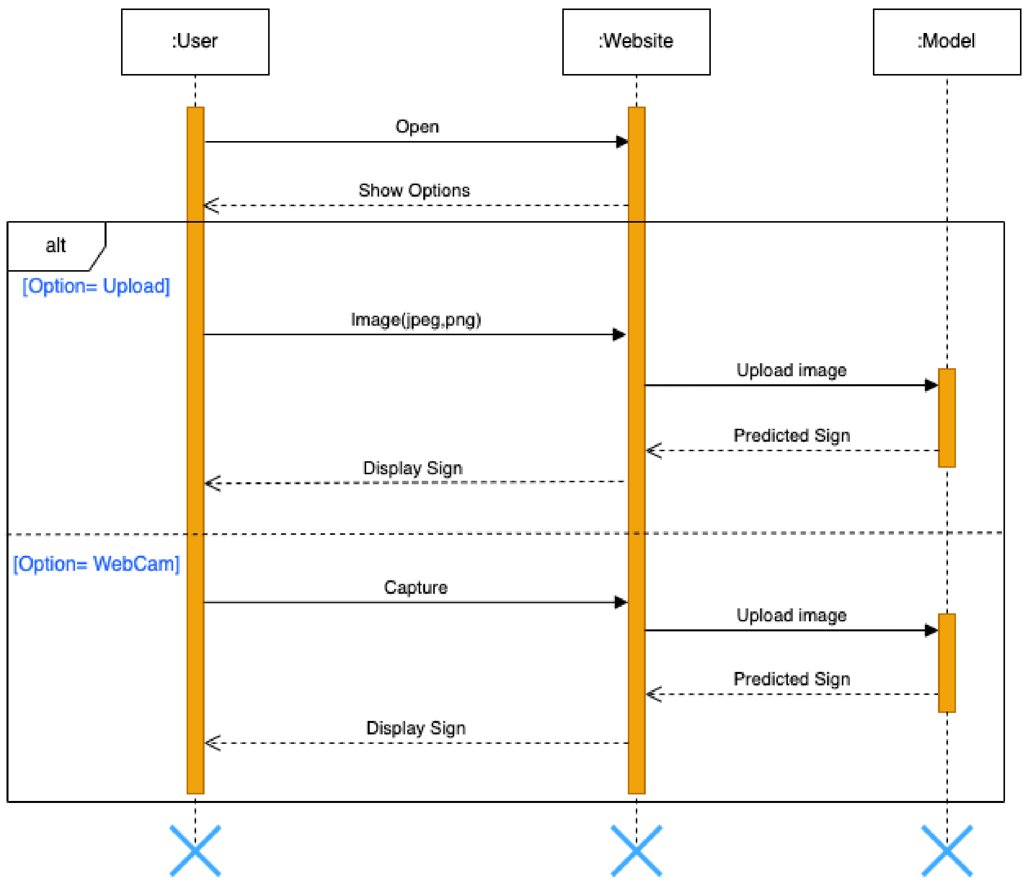


Figure 10 – Sequence Diagram

5.3 Use Case Diagram:

A use case chart is a type of behavioural graph created because of a Use-case analysis. Its goal is to show a graphical representation of the utility provided by a framework in terms of performers, their goals (referred to as utilisation cases), and any conditions that exist between those usage instances. The data on how customers and use cases are related to the framework is shown in the use case chart. To demonstrate the framework's utility, use cases are employed during the elicitation and assessment of requirements. The focus of use cases is on the framework's operation from the outside.

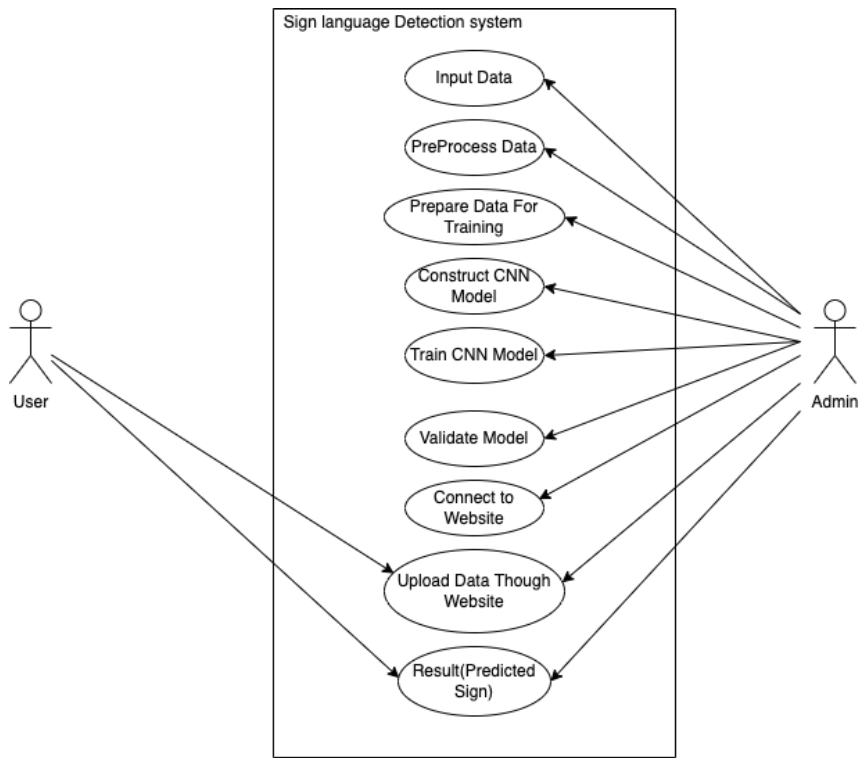


Figure 11 – Use Case Diagram

6 Methodology and Implementation

This part will go through how the project was put together, what development approaches were used, and how the entire work was broken down into smaller chunks to reduce development time. It will also go through any issues that arose during the development process.

6.1 Methodology

Because it was a solo project, an agile development style worked well. This allowed the tasks to be broken down into many discrete subsections that could be operated on independently and iteratively before being integrated to form the overall answer.

There are four essential tasks: data set creation, CNN model creation and implementation, real-time detection system implementation, and website creation and connection. All these things might be worked on independently and then integrated later. This allowed the real-time detection system to be used to evaluate a single CNN design.

When developing the real-time model for the website, an iterative method was used, allowing for the addition of new features and issue fixes as the project progressed. This iterative strategy proved effective, but it was constrained by the amount of time available, as ideas might emerge at any time.

6.2 Implementation

This section is separated into four sections, each of which will illustrate how each piece was completed independently. This section demonstrates how each component was built and how it all works. Any obstacles or challenges encountered will be discussed here as well.

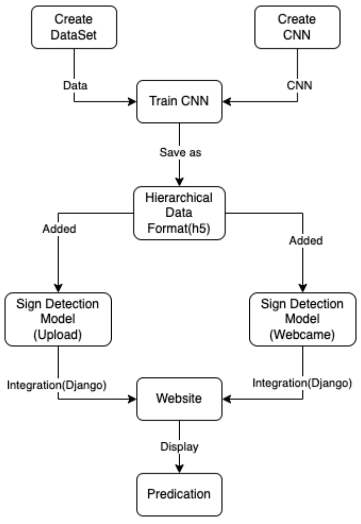


Figure 12 – Development Path

Figure 12 represents the path taken during the project's development, with the beginning of the project at the top and the finish at the bottom. It can be observed that portions might be worked on separately; for example, data could be organised concurrently with the creation of the real-time face detector and CNN architecture could be implemented together with the training of the CNN for mask detection.

6.2.1 Dataset

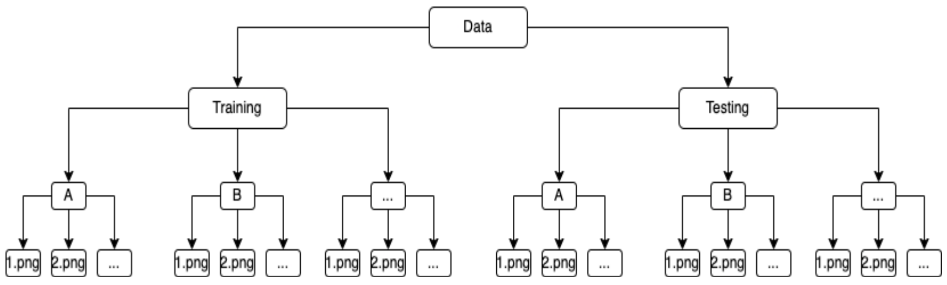


Figure 13 – Dataset Architecture

This dataset has a total of 52,000 images of signs(alphabets); the dataset is separated into two sections: training and testing. The testing dataset has a capacity of 6,500 images, and the training dataset has a total of 45,000 images. In both the training and testing dataset, the images are being classified into 26 categories, each category images labelled with alphabets from A to Z. They are 26 labelled images folders in both the training and testing dataset. In each alphabet labelled folder, they are 250 images of that alphabet in the testing dataset and 1,750 images of that alphabet in the training dataset.

The reason behind creating a dataset is that I didn't find a massive dataset of sign language on the internet; they are few datasets available on the internet which are small and without masking; if data is small, the accuracy of the model will be decreased. The main goals of this project are to get good accuracy which makes prediction more accurate and to apply masking. I implemented CNN on a dataset from the internet, then the accuracy was very low when CNN was trained, and the model predicted wrong results due to the lack of images and the noise created by the background of photos. They are many reasons that affect the accuracy of the model while predicting, like skin colour, background, and lighting conditions. To get rid of it all, I generated a dataset and used skin Filtering on the incoming video frames to detect hand motions. I wrote a python code for creating a dataset and for applying masking. From the Previous literature, as detailed in Chapter 2, maximum researchers have made their own datasets.

6.2.2 Pre-Processing

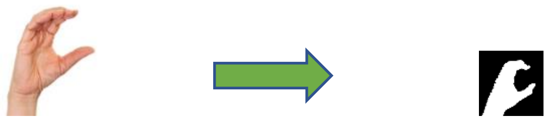


Figure 14 – Pre-processing

Every image goes through these processes skin modelling, Removal of Background, Conversion from RGB to binary and resizing as part of pre-processing.

Pre-processing is required to improve stability and identification accuracy; Data pre-processing prepares the picture series for identification. For example, prior to computing the diagonal sum and other algorithms, a pre-processing step is conducted to provide a suitable image, which is necessary for proper classification. Image enhancement necessitates image processing. During the pre-processing of an RGB image, the colour space is converted to HSV. This was done because the HSV colour scheme was less responsive to lighting changes than the RGB colour space. The data was then filtered, smoothed, and the most prominent binary linked object was evaluated, obviating the need to include skin-coloured items other than hands. Smoothing and filtering are used to achieve an excellent outcome. The purpose of picture segmentation is to locate the hand item in the image.

6.2.2.1 Skin Modelling

RGB

RGB is a three-dimensional colour space pixel in which every pixel has a set of three colours at a given position. For detecting skin regions, this approach is commonly employed in image pre-processing.

HSV (Hue, Saturation and Value)

Hue detects dominant hue, density defines colourfulness, and value determines brightness or intensity in HSV. This is sufficient for selecting a single shade, but it overlooks the intricacy of colouration. It makes a trade-off between calculation time and visual significance.

Removal of Background

Because I've noticed that backdrop has a significant impact on detection results, I've chosen to eliminate it. Instead of utilising any built-in ones, I wrote our own code for this.

Conversion from RGB to Binary

All methods take an RGB input and transform it into binary format to make it easier to recognise any gesture while preserving the brightness factor in the image.

Skin Mask

This programme converts a multicoloured picture into a binary skin mask. We employ colour components to determine whether the current pixel belongs in the skin colour space. Consequently, a binary image known as a skin mask is created.

6.2.3 Convolutional Neural Network

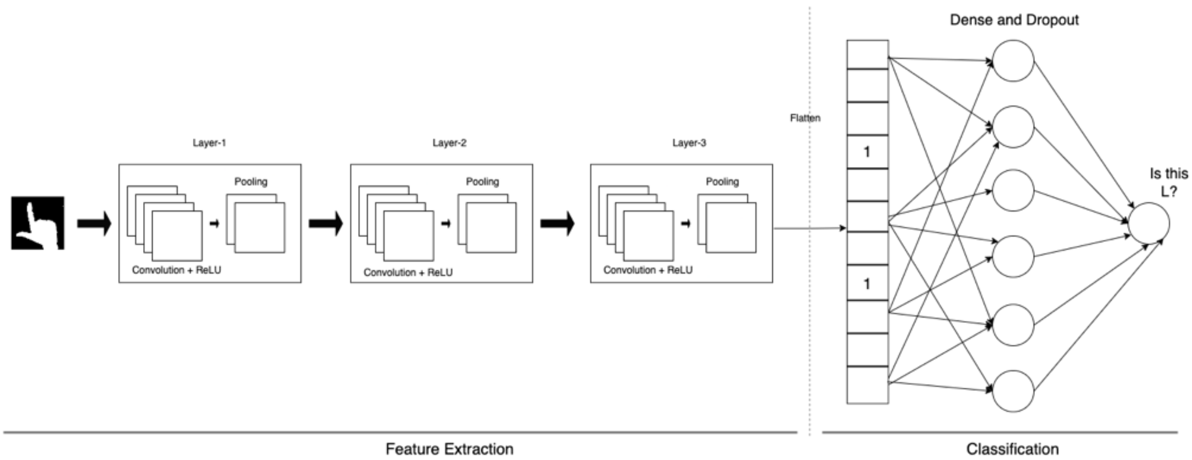


Figure 15 – CNN Architecture

The CNN layer is the most significant; it builds a convolved feature map by applying a filter to an array of picture pixels. I developed a CNN with three layers, each layer using convolution, ReLU, and pooling. Because CNN does not handle rotation and scaling by itself, a data augmentation approach was used. A few samples have been rotated, enlarged, shrunk, thickened, and thinned manually.

Convolution filters are applied to the input using 2D Convolutions in order to extract the most significant characteristics. The kernel glides in two dimensions in 2D convolution, which exactly suits the spatial properties. Convolution sparsity, when used with pooling for location invariant feature detection and parameter sharing, lowers overfitting.

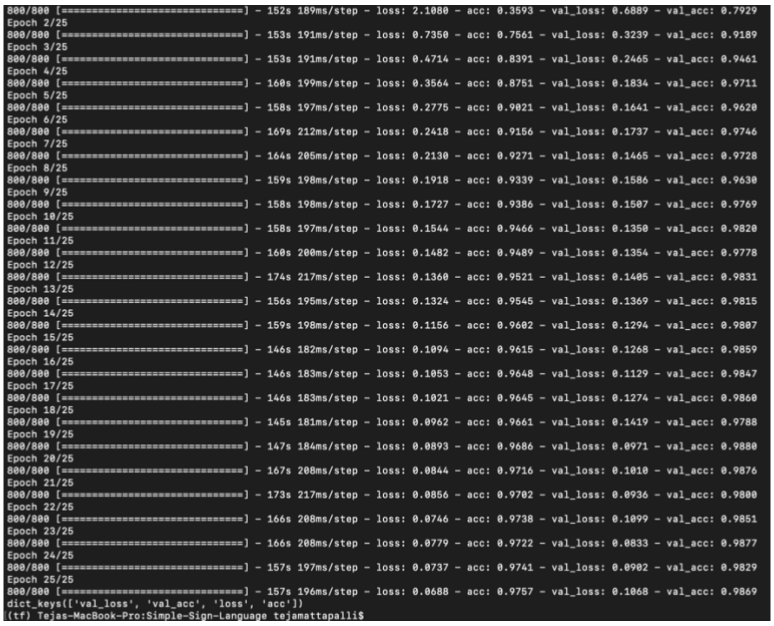
ReLU layer is a layer where data travels through each layer of the network, the ReLU layer functions as an activation function, ensuring non-linearity. Without ReLU, the dimensionality that is desired would be lost. It introduces non-linearity, accelerates training, and reduces computation time.

A pooling layer is a layer that gradually decreases the dimension of the feature and variation of the represented data. Decreases dimensions and computation, speed up processing by reducing the number of parameters that the network must compute, reduces overfitting by reducing the number of parameters, and makes the model more tolerant of changes and distortions. Pooling strategies include max pooling, min pooling, and average pooling; I tried max pooling. The maximum input of a convolved feature is used in max pooling.

Flatten is used to transform the data into a one-dimensional array for input to the next layer.

Dense the weights are multiplied by a matrix-vector multiplication of the input tensors, followed by an activation function. Apart from an activation function, the essential argument that we define here is units, which is an integer that we use to select the output size.

Dropout layer is a regularisation approach that eliminates neurons from layers at random, along with their input and output connections. As a consequence, generality is improved, and overfitting is avoided.



```
800/800 [==============================] - 152s 189ms/step - loss: 2.1080 - acc: 0.3593 - val_loss: 0.6889 - val_acc: 0.7929
Epoch 2/25
800/800 [==============================] - 153s 191ms/step - loss: 0.7350 - acc: 0.7561 - val_loss: 0.3239 - val_acc: 0.9189
Epoch 3/25
800/800 [==============================] - 153s 191ms/step - loss: 0.4714 - acc: 0.8391 - val_loss: 0.2465 - val_acc: 0.9461
Epoch 4/25
800/800 [==============================] - 160s 199ms/step - loss: 0.3564 - acc: 0.8751 - val_loss: 0.1834 - val_acc: 0.9711
Epoch 5/25
800/800 [==============================] - 158s 197ms/step - loss: 0.2775 - acc: 0.9021 - val_loss: 0.1641 - val_acc: 0.9620
Epoch 6/25
800/800 [==============================] - 169s 212ms/step - loss: 0.2418 - acc: 0.9156 - val_loss: 0.1737 - val_acc: 0.9746
Epoch 7/25
800/800 [==============================] - 164s 205ms/step - loss: 0.2130 - acc: 0.9271 - val_loss: 0.1465 - val_acc: 0.9728
Epoch 8/25
800/800 [==============================] - 159s 198ms/step - loss: 0.1918 - acc: 0.9339 - val_loss: 0.1586 - val_acc: 0.9630
Epoch 9/25
800/800 [==============================] - 158s 198ms/step - loss: 0.1727 - acc: 0.9386 - val_loss: 0.1507 - val_acc: 0.9769
Epoch 10/25
800/800 [==============================] - 158s 197ms/step - loss: 0.1544 - acc: 0.9466 - val_loss: 0.1350 - val_acc: 0.9820
Epoch 11/25
800/800 [==============================] - 160s 200ms/step - loss: 0.1482 - acc: 0.9489 - val_loss: 0.1354 - val_acc: 0.9778
Epoch 12/25
800/800 [==============================] - 174s 217ms/step - loss: 0.1360 - acc: 0.9521 - val_loss: 0.1405 - val_acc: 0.9831
Epoch 13/25
800/800 [==============================] - 156s 195ms/step - loss: 0.1324 - acc: 0.9545 - val_loss: 0.1369 - val_acc: 0.9815
Epoch 14/25
800/800 [==============================] - 159s 198ms/step - loss: 0.1156 - acc: 0.9602 - val_loss: 0.1294 - val_acc: 0.9807
Epoch 15/25
800/800 [==============================] - 146s 182ms/step - loss: 0.1094 - acc: 0.9615 - val_loss: 0.1268 - val_acc: 0.9859
Epoch 16/25
800/800 [==============================] - 146s 183ms/step - loss: 0.1053 - acc: 0.9648 - val_loss: 0.1129 - val_acc: 0.9847
Epoch 17/25
800/800 [==============================] - 146s 183ms/step - loss: 0.1021 - acc: 0.9645 - val_loss: 0.1274 - val_acc: 0.9860
Epoch 18/25
800/800 [==============================] - 145s 181ms/step - loss: 0.0962 - acc: 0.9661 - val_loss: 0.1419 - val_acc: 0.9788
Epoch 19/25
800/800 [==============================] - 147s 184ms/step - loss: 0.0893 - acc: 0.9686 - val_loss: 0.0971 - val_acc: 0.9880
Epoch 20/25
800/800 [==============================] - 167s 208ms/step - loss: 0.0844 - acc: 0.9716 - val_loss: 0.1010 - val_acc: 0.9876
Epoch 21/25
800/800 [==============================] - 173s 217ms/step - loss: 0.0856 - acc: 0.9702 - val_loss: 0.0936 - val_acc: 0.9800
Epoch 22/25
800/800 [==============================] - 166s 208ms/step - loss: 0.0746 - acc: 0.9738 - val_loss: 0.1099 - val_acc: 0.9851
Epoch 23/25
800/800 [==============================] - 166s 208ms/step - loss: 0.0779 - acc: 0.9722 - val_loss: 0.0833 - val_acc: 0.9877
Epoch 24/25
800/800 [==============================] - 157s 197ms/step - loss: 0.0737 - acc: 0.9741 - val_loss: 0.0902 - val_acc: 0.9829
Epoch 25/25
800/800 [==============================] - 157s 196ms/step - loss: 0.0688 - acc: 0.9757 - val_loss: 0.1068 - val_acc: 0.9869
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
(tf) Tejas-MacBook-Pro:Simple-Sign-Language tejamattapalli$
```

Figure 16 – Trained CNN Model

Figure22 shows the training of the model; it took 4 hours of time to train the model for 25 epochs. Recorded accuracy is 97.57%, loss is 6.88%, validation accuracy is 98.69% and validation loss is 10.68%.

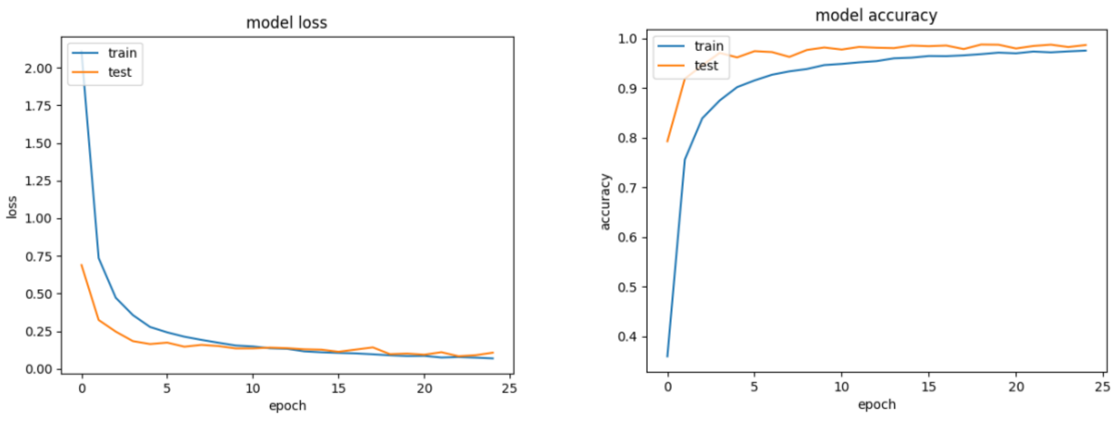


Figure 17 – Model Loss and Accuracy

As seen in figure 17, The training accuracy continued to improve after four epochs of operating a network for 25 epochs, but the test accuracy stopped improving; similar behaviour was found with a few different structured models that we investigated. That meant the model had begun to overfit.

As a result, each network generated and evaluated by the Tuner was run for up to 8 epochs, as that was the best middle ground of the network continuing to learn but neither over or underfitting, guaranteeing that the reported accuracy was true. Our output layer was a Dense layer with a relu and softmax activation function at the end of the network. Make a multinomial probability distribution prediction. That is, softmax is employed as the activation function in multi-class classification issues when class membership on more than two class labels is needed. This CNN model has achieved 97.57% of accuracy, which is a good percentage of accuracy; from previous literature, as detailed in Chapter 2, Salian and their team used Haar Cascade instead of CNN, and he achieved 92.88% accuracy; Haar Cascade is a prevalent pre-trained model, It is an Object Detection Algorithm that is used to recognise faces in images or real-time video, and I can't use Haar cascade for my model, This algorithm uses edge/line detection features. As detailed in Chapter 2, Rao and their team have designed and tested different CNNs, the highest accuracy they achieved is 92.88%.

6.2.4 Detection Model

6.2.4.1 Through Web Cam

The webcam that is linked to every gadget on the market today may be used to detect signs in real-time by granting authorisation to the user to access the webcam through the browser. The webcam sign detection comprises many components that aid in the prediction of the letter that the user displays in the green outline box-defined region.

Inner Workings

Two URLs are used to display and extract the mask layer from the webcam.

- /video: This is the URL for capturing and streaming the image response in bytes, which is updated every frame. The StreamingHttpResponse is updated using the gencamera class, which initiates, gets the frame, masks, and returns the StreamingHttpResponse that is updated using the VideoCamera class, which provides the byte form for the picture.

- /video feed: This is the default page of the webcam prediction, which is connected to the web application's homepage. It's made up of the image tag, which is used to show the multipart-x/data that the /video returns. As the stream is processed, the picture source is updated with a new frame.

Data flow

The Camera class, which originates and analyses the photographs, captures the User frames. The OpenCV library's [18] Video Capture function is used to capture the video. It requires the camera's index, which is set to zero by default. The camera feed is used to initialise the object. After the user's vision has been restored, the web camera is restarted to pre-process, store, forecast, and eventually show the rebuilt video frame.

- Get frame: this method is used to obtain a frame from the video instance in the previous section. The mask that is applied to the camera feed in the right corner is received by the frame. Bytes from the JPEG picture format are used to return the frame.

- Pre-process: takes as an argument the frame returned by the previous method. To begin, the image is flipped to rectify the webcam's mirror effect. The user will be aided in showing the motions in the box by defining the bounding box. The picture inside the box is cropped out of the frame and then utilised to apply different filters on the cropped image to get the desired masking. The HSV filter is one of the filters. To filter all three (R, G, and B) channels in the picture NumPy array, the filter must be applied in the range of arrays. The higher and lower boundaries are named upper blue and lower blue, respectively. It's made up of constants like u_s, u_h, and u_v for the upper. Similarly, the lower bound contains the l_h, l_v variables, but it's worth noting that l_s is a dynamic variable that changes depending on the illumination and contrast of the environment to help the user receive the best forecast in any situation.

- The lighting adjuster is a slider that represents the l_s dynamic variable. By default, the slider is set at 55 units. The value of the slider is given to the controller in the form of POST data from the input to the URL once it is updated. If the method is POST, the value is modified to reflect the global importance and passed back to the view. Following the same reasoning as before, but with the updated variable, it results in a better prediction that may be altered based on the mask feed on the corner to discover the letter more quickly.

- Predictor: the single argument is the prediction from the trained model. The picture that was previously saved as 1.png for every frame masking image by get frame() is loaded and turned into a NumPy array. The picture array is loaded into the ".h5" format TensorFlow [17] machine learning model. The Keras [19] package and the load model function defined by the package are used to do this. The letter may be predicted from a case with style conditionals based on the output from the model, where the indexing of the resultant array is used to identify the letter of the alphabet that is predicted and returned.

The letter is then inserted into the frame using the putText method, and the frame is finished and ready to be shown to the user.

6.2.4.2 Manually Upload

Sign Detection may also be performed on a pre-masked image to extract the sign represented in the picture. This is important for creating a group of photographs that may be used to educate or predict using images from a dataset.

- /upload: This URL is used to show the input form for uploading an image file, as well as the prediction that is sent to the upload view.

- Models.py: The Model is built using the Image field to show the input field, which can only accept JPEG and PNG image types and cannot be false.

- Forms.py: To accept the image and predict the letter using the TensorFlow [17] model, the form is generated using the model above, and the save function in the forms class. The structure is constructed from the fields in the model when the object is formed.

- sav(): This function is used to react when an image has been uploaded to a web application. This function of the Form Object may be called using the response object from the controller's request when the process is called. The Keras [19] model is imported from the project directory, which has already been pre-compiled to offer picture prediction. The stored image is grabbed from the leading directory and turned into a three-channel NumPy array. Because the image you've submitted has already been pre-processed. The model is given the array, and the Keras [19] model returns the prediction.

- gesture_view: The controller shows the HTML page upload in gesture view. The Form object is constructed and then supplied to the page's context. The Django [16] Template is used to represent the form as tags on the page, which are presented as a paragraph. An upload button serves as a substitute for the submit button on the page. When the POST request has been sent. To distinguish between GET and POST requests, the conditional statement is employed. Instead of using the form object to get the prediction, the prediction character is delivered, which is shown to the user using conditional tags in Django [16] Template.

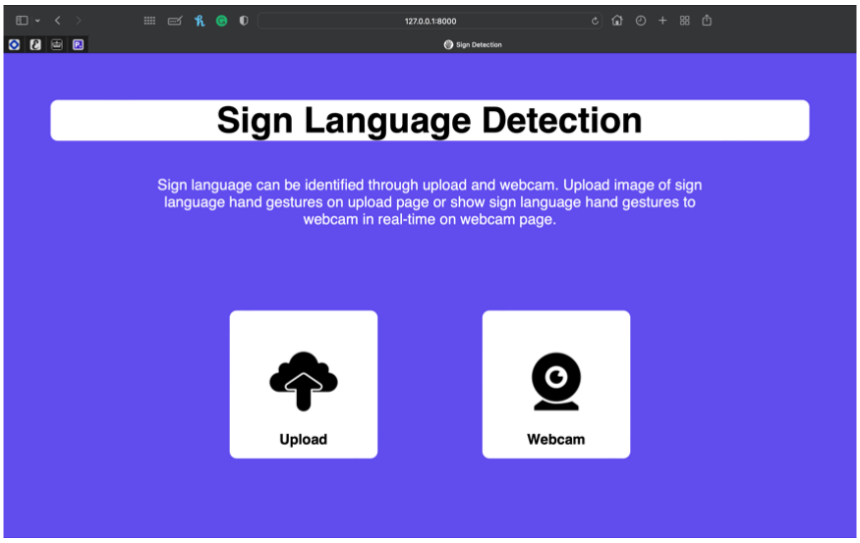
- Predictor: the single argument is the prediction from the trained model. The picture that was previously saved using sav() is now turned into a NumPy array. The picture array is loaded into the.h5 format TensorFlow [17] machine learning model. The Keras [19] package and the load model function defined by the package are used to do this. The letter may be predicted using case-switch style conditionals based on the output from the model, where the indexing of the resultant array is used to indicate the letter of the alphabet that is predicted and returned.

6.2.4.3 Website

The website is developed using HTML and CSS for the frontend and python for backend; the Django framework is used to connect both the frontend and backend. The website contains a total of three pages home page, upload page and webcam page. The website is designed in a way that users can easily understand the purpose of the website and can easily use it. From previous literature, as detailed in Chapter 2, I didn't find any resource regarding detection websites where they are linking machine learning models to websites.

Home Page

The dashboard is linked with two pages upload page, and a webcam page; Sign language can be identified through both upload and webcam pages. This home page had information to guide users on how to use the website, as you can see below the image.



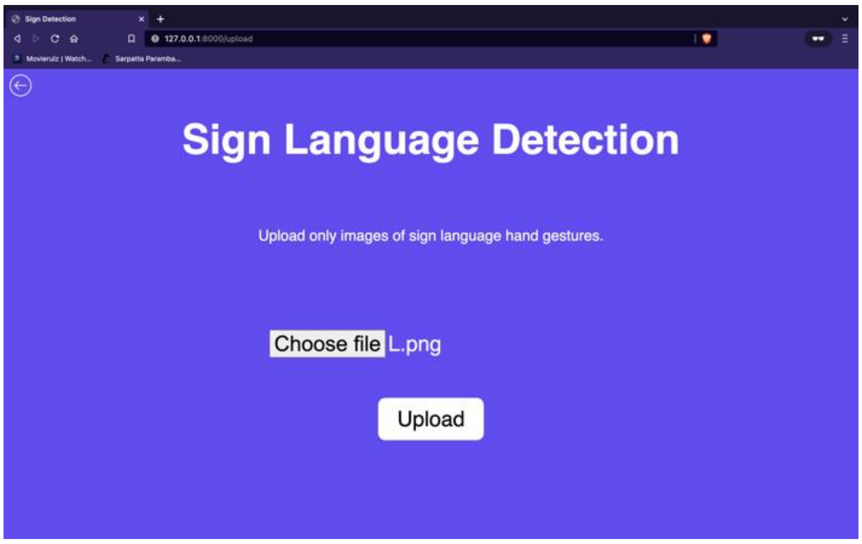
Sign Language Detection

Sign language can be identified through upload and webcam. Upload image of sign language hand gestures on upload page or show sign language hand gestures to webcam in real-time on webcam page.

Figure 18 – Website (Home Page)

Upload Page

It is a page where users can upload images of sign language hand gestures manually from their local storage, and predicted signs will be displayed on the same screen, and users can upload as many as images; every image will be deleted after the model predicts the sign.



Sign Language Detection

Upload only images of sign language hand gestures.

Figure 19– Website (Upload Page)

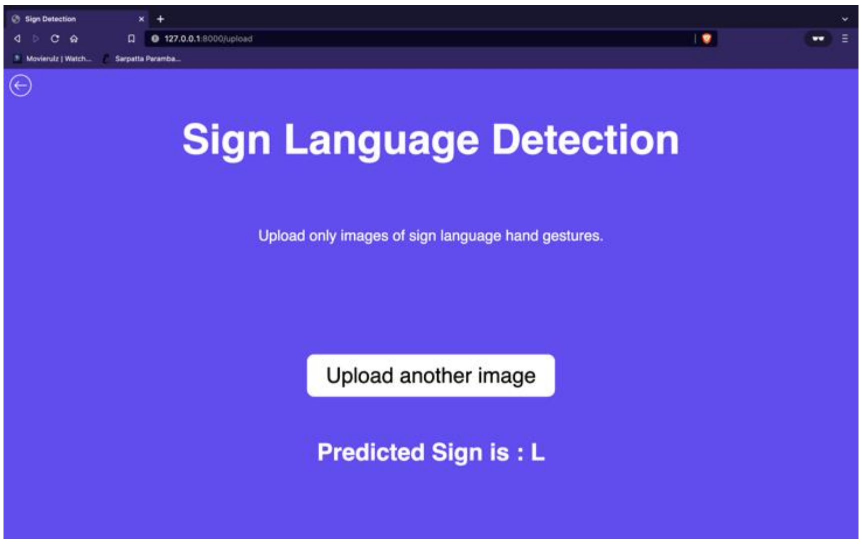


Figure 20– Website (Result Page)

Webcam Page

It is a page where users can show sign language hand gestures to the webcam in the green box displayed on the screen; masking will be displayed at the right corner, and predicted sign will be displayed at the top left corner. A masking adjuster can be used to adjust making based on lighting conditions. The whole prediction and masking will perform on a live image through a webcam. All photos collected through the webcam will go through masking, which makes the model truly anonymised, and they will be deleted after prediction.

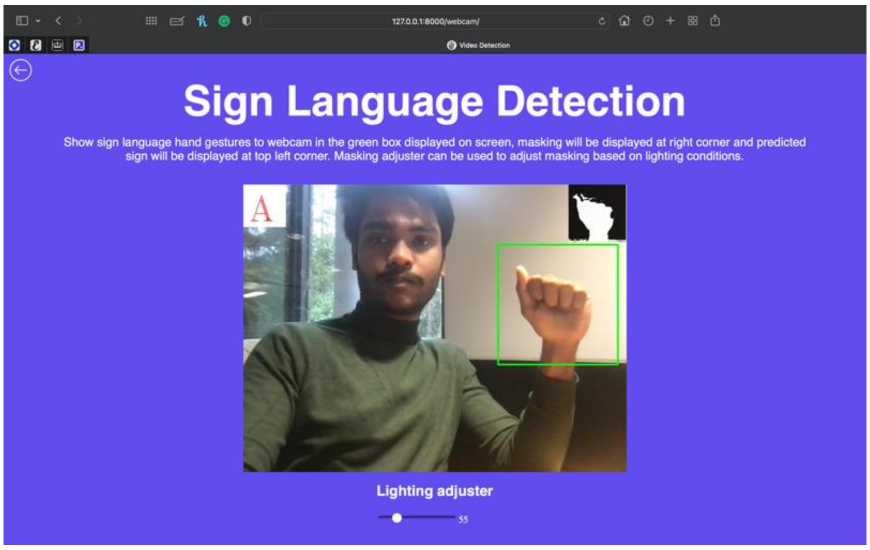

127.0.0.1:8000/webcam/
Video Detection
Sign Language Detection
Show sign language hand gestures to webcam in the green box displayed on screen, masking will be displayed at right corner and predicted sign will be displayed at top left corner. Masking adjuster can be used to adjust masking based on lighting conditions.
A
Lighting adjuster
55

Figure 21 – Website (Webcam Page)

7 Evaluation Strategy

This section will go through how well the model will be assessed. It is critical to analyse every model element that led to achieving the efficiency of sign language recognition.

7.1 Evaluation

7.1.1 Accuracy Analysis Using Confusion Matrix

To assess our classification findings, I will utilise a confusion matrix table. This is significant since accuracy is not a credible criterion for testing the actual performance of an image recognition algorithm. The most frequent performance indicators used to determine classification efficacy are recall, accuracy, F1 score, and precision [12]. Parameters are used to assess the performance. FP, TP, FN, TN.

- False Positive = The incorrect guesses made on a negative arriving.
- False Negative = The incorrect guesses made on a positive instance.
- True Negative = The number of correct predictions for a positive occurrence is denoted.
- True Positive = The number of correct predictions for a negative occurrence is denoted.

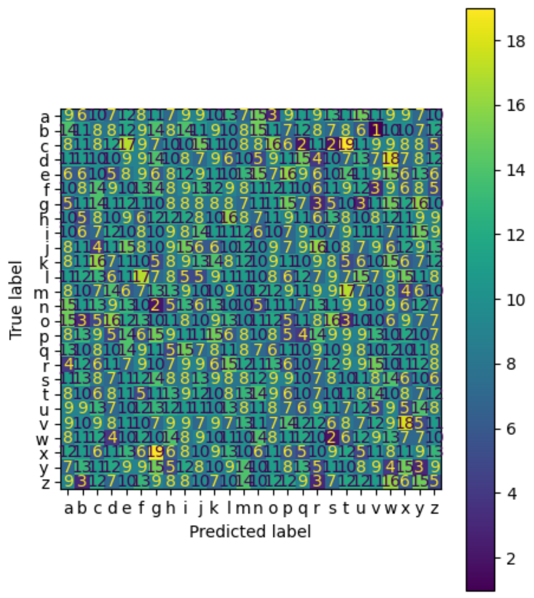


Figure 22 – Confusion Matrix

Accuracy can be determined using this equation:

$$\boxed{Accuracy = \frac{T.N + T.P}{T.P + T.N + F.P + F.N}}$$

Figure 23 – Accuracy Formula [12]

After the calculations, we get 0.9757 which is around 98% accuracy.

7.2 Testing

7.2.1 Unit Testing

Unit testing is a technique where the individual programming pieces are tested separately, known as units [25]. Each unit is checked to make sure everything is working correctly. Every unit is examined independently from the rest of the system. After testing the model, the results were compared to what was predicted. MODULE TESTING is another name for Unit Testing. This testing is done in steps, and every unit performed satisfactorily in terms of the module's intended output. Units are Dataset Creation, Training CNN, Trained mode, Detection Model(upload), Detection Model (Webcam), Website (Home Page), Website (Upload Page) and Website (Webcam Page).

Note: All screenshots are available in the Appendix.

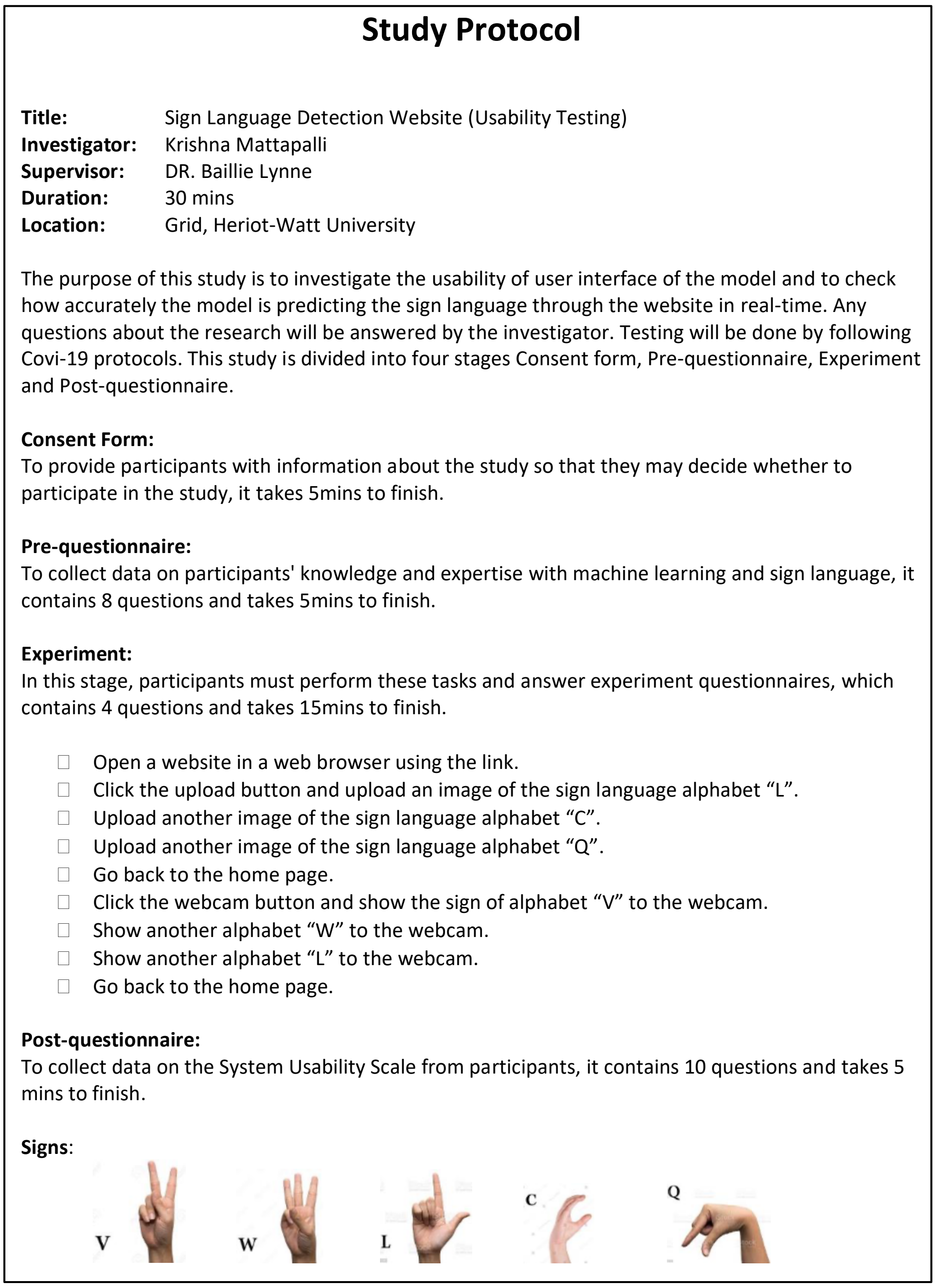
7.2.2 Usability Testing

A group of users were asked to utilise the website independently while it was being evaluated. The major purpose was to evaluate the results of the sign classifier model and collect data on the user's experience. To do this, a testing technique was developed.

7.2.2.1 Testing Protocol

To test the system, a pre-questionnaire, experiment questionnaire and post-questionnaire were created. Users were asked if they had any prior knowledge of machine learning and sign language or any experience with machine learning websites/apps in the pre-questionnaire. They were then instructed to open and utilise the website. During the testing phase, the results of the sign accuracy were recorded. Finally, in the post-questionnaire, the SUS (System Usability Scale) Framework was employed, and users were requested to complete it.

Note: Consent form is available in the Appendix.



Study Protocol

Title: Sign Language Detection Website (Usability Testing)
Investigator: Krishna Mattapalli
Supervisor: DR. Baillie Lynne
Duration: 30 mins
Location: Grid, Heriot-Watt University

The purpose of this study is to investigate the usability of user interface of the model and to check how accurately the model is predicting the sign language through the website in real-time. Any questions about the research will be answered by the investigator. Testing will be done by following Covi-19 protocols. This study is divided into four stages Consent form, Pre-questionnaire, Experiment and Post-questionnaire.

Consent Form:
To provide participants with information about the study so that they may decide whether to participate in the study, it takes 5mins to finish.

Pre-questionnaire:
To collect data on participants' knowledge and expertise with machine learning and sign language, it contains 8 questions and takes 5mins to finish.

Experiment:
In this stage, participants must perform these tasks and answer experiment questionnaires, which contains 4 questions and takes 15mins to finish.

- ☐ Open a website in a web browser using the link.
- ☐ Click the upload button and upload an image of the sign language alphabet "L".
- ☐ Upload another image of the sign language alphabet "C".
- ☐ Upload another image of the sign language alphabet "Q".
- ☐ Go back to the home page.
- ☐ Click the webcam button and show the sign of alphabet "V" to the webcam.
- ☐ Show another alphabet "W" to the webcam.
- ☐ Show another alphabet "L" to the webcam.
- ☐ Go back to the home page.

Post-questionnaire:
To collect data on the System Usability Scale from participants, it contains 10 questions and takes 5 mins to finish.

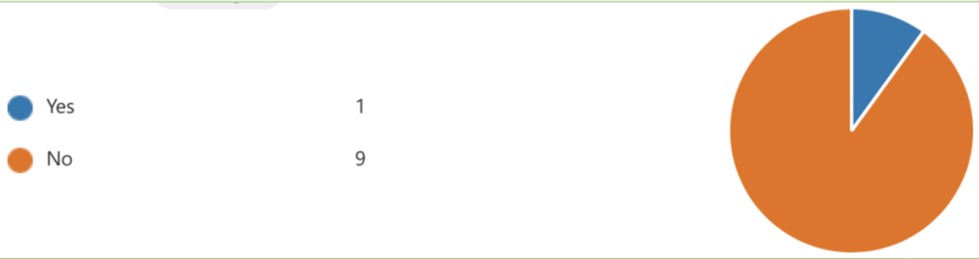
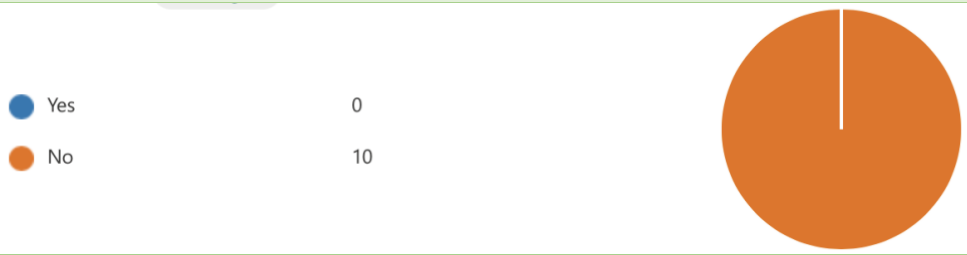
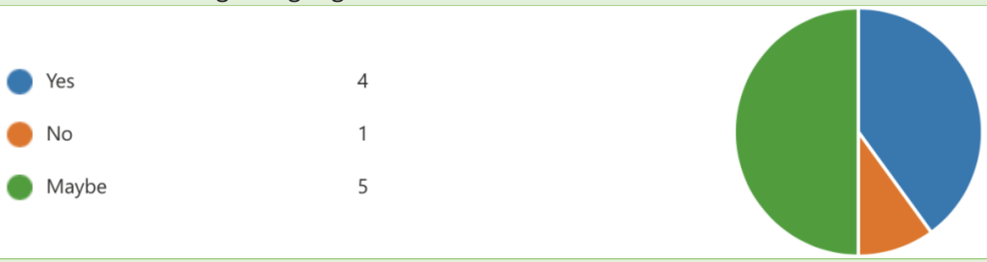
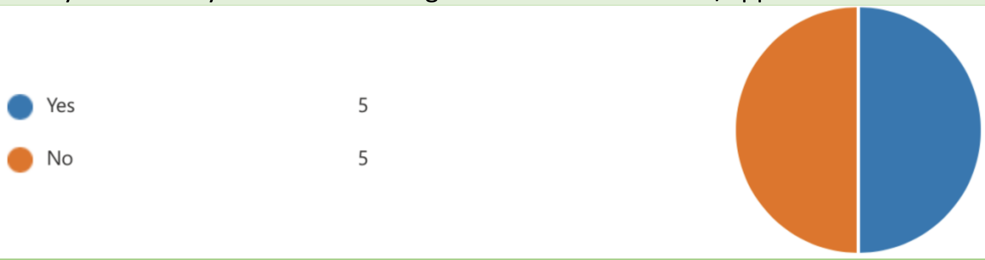
Signs:

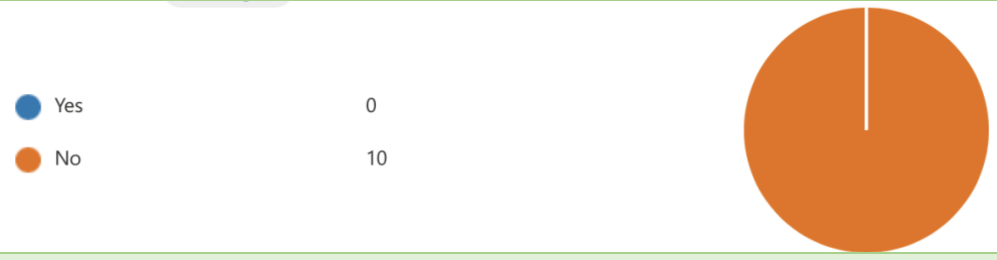
Figure 24- Usability Testing Protocol

7.2.2.2 Questionnaire and Results

Pre-questionnaire

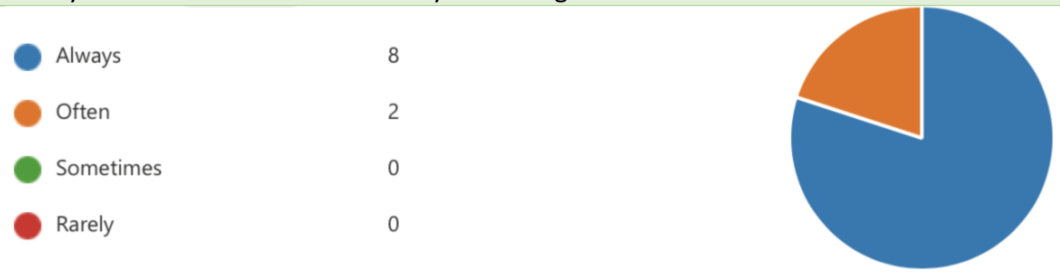
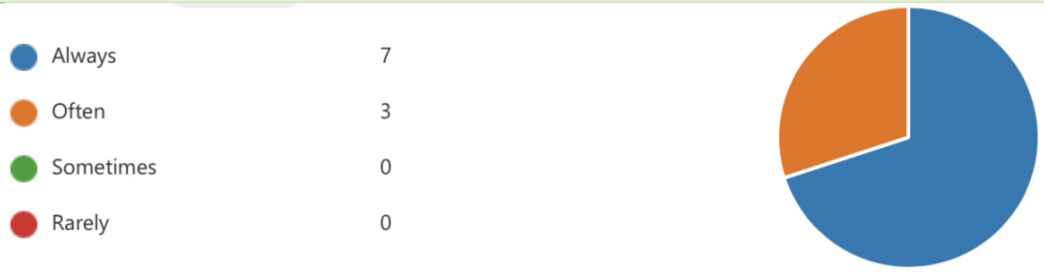
A preliminary survey was conducted to learn more about the consumers' expertise and experience with machine learning-based websites. The questions and results obtained are listed in the table below.

No.	Results of the questions/responses
Q1	Do you know any sign language?
A1	
Q2	Can you understand and communicate in sign language?
A2	
Q3	If you answered "No" to the preceding question, have you felt it is a deficit to communicate in sign language?
A3	
Q4	Do you know what the phrase Machine Learning means?
A4	Six participants answered yes, 3 participants answered AI, and 1 participant said they didn't know anything.
Q5	Have you used any Machine Learning identification websites/apps?
A5	

Q6	If you answered "Yes" to the preceding question, did the website/app use machine learning to detect Sign Language?
A6	 ● Yes 0 ● No 10
Q7	If you answered "Yes" to the preceding question, please provide a brief description of your experience below.
A7	No results were collected here.
Q8	How essential do you believe sign language detection websites to be in the future?
A8	Seven participants answered it is essential, 1 participant answered it is necessary, and 1 participant answered it is necessary.

Experiment questionnaire

This survey was devoted to user experience questions and website accuracy prediction; the questions and results are listed in the table below.

No.	**Results of the questions/responses**
Q1	Were there any challenges with the website while opening and using it?
A1	During the opening and use, none of the participants had any difficulties.
Q2	On the website, you saw a prediction of sign language labels after uploading an image locally. How often were the accuracy results right?
A2	 ● Always 8 ● Often 2 ● Sometimes 0 ● Rarely 0
Q3	On the website, you see a prediction of sign language labels in real-time through a webcam. How often were the accuracy results right?
A3	 ● Always 7 ● Often 3 ● Sometimes 0 ● Rarely 0
Q4	Do you believe the UI (User Interface) was straightforward and straightforward to use?
A4	The user interface was rated as straightforward and easy to use by 8 out of 10 individuals.

post-questionnaire

The website's usability and functionality were assessed using a SUS (System Usability Scale). The questions and results are listed in the table below. 1 strongly disagrees, 2 disagrees, 3 neutral, 4 agree, and 5 strongly agree.

	Statements	1	2	3	4	5
Q1	I think that I would like to use this website frequently.	0	0	1	3	7
Q2	I found the website unnecessarily complex.	9	1	0	0	0
Q3	I thought the website was easy to use.	0	0	0	2	8
Q4	I think that I would need the support of a technical person to be able to use this website.	7	3	0	0	0
Q5	I found the various functions in this website were well integrated.	0	0	3	1	6
Q6	I thought there was too much inconsistency in this website.	7	2	1	0	0
Q7	I would imagine that most people would learn to use this website very quickly.	0	0	1	1	8
Q8	I found the website very cumbersome to use.	8	2	0	0	0
Q9	I felt very confident using the website.	0	1	0	2	7
Q10	I needed to learn a lot of things before I could get going with this website.	6	1	1	1	1

System Usability Scale

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
SUS Points	2.9	3.5	3.1	4.0	4.0	3.1	4.0	4.0	3.6	3.7

Total SUS score: 35.9*2.5 = 89.75.

System rates an ‘A' grade on the System Usability Scale by scoring approximate 90 points.

8 Conclusion

This chapter will summarise the whole project by analysing the successes that were and were not accomplished, the limits that this project experienced, and future work that may be done to improve and extend the project's capabilities.

The purpose of this research was to develop a real-time sign language detection system that could be used on the web. Previous literature, as detailed in Chapter 2, has previously developed sign language classifiers using CNNs, but does not address the models' real-time classification efficacy on a website/application. Although this aim was met in general, its usefulness in other sectors can be questioned. The overall accuracy the model achieved is 0.9757, around 98%, which is pretty good. A massive dataset with 52,000 images was created for this project. A lot of pre-processing was done, and for feature extraction, masking was applied on each end of every image of the dataset. The dataset was trained using CNN, and the trained model has been used for creating the detection system for both models through webcam and manually upload. Using HTML and CSS, the front end was developed, and python was used for the back end; for connecting the front end and back end, I used the Django framework. The final website is fulfilling the aim and objectives. The model performed well in usability testing without having any issues while the experiment was going; the model achieved 89.75 points in SUS scour.

8.1 Limitations

- I couldn't find an extensive sign language dataset.
- Didn’t find any pre-trained model for identification.
- More signs were currently being created. However, towards the end of the project, I did not have enough time to adequately implement this approach. The majority of my time was spent studying and practising image recognition models.
- Masking adjustment could be don’t in a better way, like automatic masking based on light conditions rather than manually adjustment.

8.2 Future Work

- The website could be improved. The primary focus right now is on getting the model to run and perform. As a result, the website was made to be relatively simple. The software might, however, be modified in the future with a more appealing look.
- The model applies masking to every image. The lighting conditions are different every ware, so users must adjust masking manually on the website, which can be done by the model by itself in future.
- The model collects images from only the green box on the screen; then, it applies making and prediction on the image. But the model could be improved by identifying the hand from any ware on the screen.
- Using this model, a mobile application can be developed, which makes it so convenient for users to use.
- There are still several alternatives for further research, such as noise reduction, dataset collection, enhanced feature extraction, or the use of a better model.

9 References

[1] 'Neamat El Gayar. (2021), Overview of Data mining and Machine Learning'. https://canvas.hw.ac.uk/courses/5291/files/799447?wrap=1 (accessed Nov. 15, 2021).

[2] Chen, Li, Song Wang, Wei-liang Fan, Jun Sun and Satoshi Naoi. "Beyond human recognition: A CNN-based framework for handwritten character recognition." *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015): 695-699.

[3] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." *arXiv preprint arXiv:1511.08458* (2015).

[4] D. Liu, 'A Practical Guide to ReLU', *Medium*, Nov. 30, 2017. https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7 (accessed Nov. 20, 2021).

[5] 'CS231n Convolutional Neural Networks for Visual Recognition'. https://cs231n.github.io/convolutional-networks/ (accessed Nov. 20, 2021).

[6] 'How do Convolutional Neural Networks work?' Url:https://e2eml.school/how_convolutional_neural_networks_work.html (accessed Oct. 20, 2021).

[7] Rao, G. Anantha, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry. "Deep convolutional neural networks for sign language recognition." In *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, pp. 194-197. IEEE, 2018.

[8] Salian, Shashank, Indu Dokare, Dhiren Serai, Aditya Suresh, and Pranav Ganorkar. "Proposed system for sign language recognition." In *2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*, pp. 058-062. IEEE, 2017.

[9] J.R.Quinlan, Induction of Decision Trees, Machine Learning 1: 81-106, 1986. Boston, Kluwer Academic Publishers

[10] Elkan, Charles. "Nearest neighbor classification." *elkan@ cs. ucsd. edu||,|| January* 11 (2011): 3.

[11] Dabre, Kanchan, and Surekha Dholay. "Machine learning model for sign language interpretation using webcam images." In *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp. 317-321. IEEE, 2014.

[12] Shah, F., Shah, M.S., Akram, W., Manzoor, A., Mahmoud, R.O. and Abdelminaam, D.S. (2021) 'Sign Language Recognition Using Multiple Kernel Learning: A Case Study of Pakistan Sign Language', *IEEE access*, 9, pp. 67548–67558. doi:10.1109/ACCESS.2021.3077386.

[13] Bantupalli, Kshitij, and Ying Xie. "American sign language recognition using deep learning and computer vision." In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4896-4899. IEEE, 2018.

[14] K.wagstaff, C.Cardie, S.Roger, S.Schroeld, Constrained K-means Clustering with Background Knowledge, Proceedings of the Eighteenth International Conference on Machine Learning, 2001

[15] 'Neamat El Gayar. (2021) Clustering_Intro'. https://canvas.hw.ac.uk/courses/5291/files/897095?wrap=1 (accessed Nov. 18, 2021).

[16] "Free and open-source web framework |Django". https://www.djangoproject.com/ (accessed Mar. 16, 2022).

[17] "Free and open-source software library for machine learning |Tensonflow". https://www.tensorflow.org/learn (accessed Feb. 25, 2022).

[18] "Image processing and performing computer vision tasks tool |OpenCV". https://pypi.org/project/opencv-python/ (accessed Feb. 22, 2022).

[19] "Open-source software library, for artificial neural network |Keras". https://keras.io/examples/ (accessed Mar. 15, 2022).

[20] "cloud-based Gantt chart and project planning solution |Gantt Chart". https://www.teamgantt.com (accessed Des. 15,2021)

[21] Burch, Carl. "Django, a web framework using python: tutorial presentation." *Journal of Computing Sciences in Colleges* 25, no. 5 (2010): 154-155.

[22] Ghimire, Devndra. "Comparative study on Python web frameworks: Flask and Django." (2020).

[23] Lewis, James R. "The system usability scale: past, present, and future." *International Journal of Human–Computer Interaction* 34, no. 7 (2018): 577-590.

[24] Bangor, Aaron, Philip T. Kortum, and James T. Miller. "An empirical evaluation of the system usability scale." *Intl. Journal of Human–Computer Interaction* 24, no. 6 (2008): 574-594.

[25] Runeson, Per. "A survey of unit testing practices." *IEEE software* 23, no. 4 (2006): 22-29.

Appendix

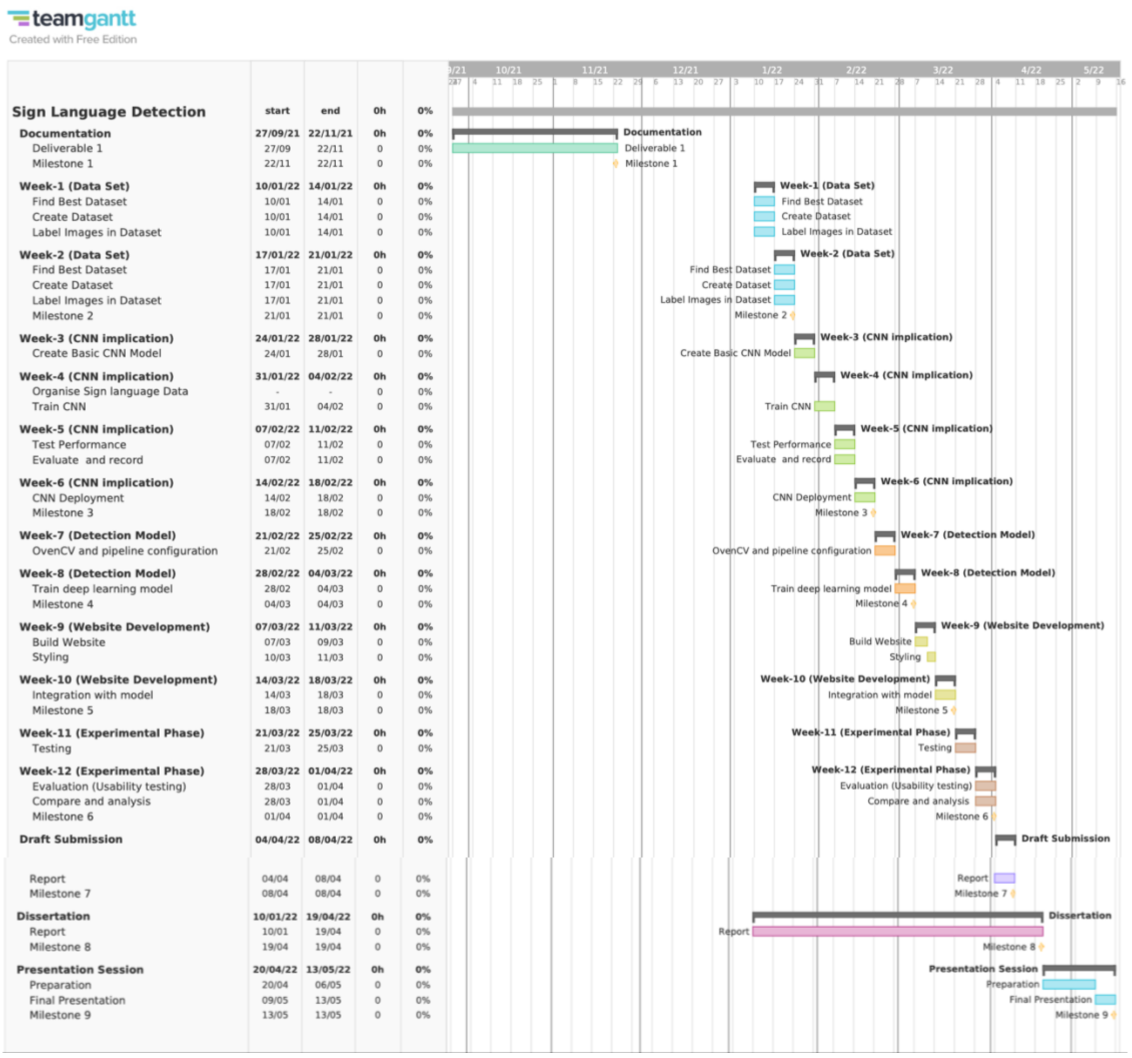

Sign Language Detection	start	end	0h	0%
Documentation	**27/09/21**	**22/11/21**	**0h**	**0%**
Deliverable 1	27/09	22/11	0	0%
Milestone 1	22/11	22/11	0	0%
Week-1 (Data Set)	**10/01/22**	**14/01/22**	**0h**	**0%**
Find Best Dataset	10/01	14/01	0	0%
Create Dataset	10/01	14/01	0	0%
Label Images in Dataset	10/01	14/01	0	0%
Week-2 (Data Set)	**17/01/22**	**21/01/22**	**0h**	**0%**
Find Best Dataset	17/01	21/01	0	0%
Create Dataset	17/01	21/01	0	0%
Label Images in Dataset	17/01	21/01	0	0%
Milestone 2	21/01	21/01	0	0%
Week-3 (CNN implication)	**24/01/22**	**28/01/22**	**0h**	**0%**
Create Basic CNN Model	24/01	28/01	0	0%
Week-4 (CNN implication)	**31/01/22**	**04/02/22**	**0h**	**0%**
Organise Sign language Data	-	-	0	0%
Train CNN	31/01	04/02	0	0%
Week-5 (CNN implication)	**07/02/22**	**11/02/22**	**0h**	**0%**
Test Performance	07/02	11/02	0	0%
Evaluate and record	07/02	11/02	0	0%
Week-6 (CNN implication)	**14/02/22**	**18/02/22**	**0h**	**0%**
CNN Deployment	14/02	18/02	0	0%
Milestone 3	18/02	18/02	0	0%
Week-7 (Detection Model)	**21/02/22**	**25/02/22**	**0h**	**0%**
OvenCV and pipeline configuration	21/02	25/02	0	0%
Week-8 (Detection Model)	**28/02/22**	**04/03/22**	**0h**	**0%**
Train deep learning model	28/02	04/03	0	0%
Milestone 4	04/03	04/03	0	0%
Week-9 (Website Development)	**07/03/22**	**11/03/22**	**0h**	**0%**
Build Website	07/03	09/03	0	0%
Styling	10/03	11/03	0	0%
Week-10 (Website Development)	**14/03/22**	**18/03/22**	**0h**	**0%**
Integration with model	14/03	18/03	0	0%
Milestone 5	18/03	18/03	0	0%
Week-11 (Experimental Phase)	**21/03/22**	**25/03/22**	**0h**	**0%**
Testing	21/03	25/03	0	0%
Week-12 (Experimental Phase)	**28/03/22**	**01/04/22**	**0h**	**0%**
Evaluation (Usability testing)	28/03	01/04	0	0%
Compare and analysis	28/03	01/04	0	0%
Milestone 6	01/04	01/04	0	0%
Draft Submission	**04/04/22**	**08/04/22**	**0h**	**0%**
Report	04/04	08/04	0	0%
Milestone 7	08/04	08/04	0	0%
Dissertation	**10/01/22**	**19/04/22**	**0h**	**0%**
Report	10/01	19/04	0	0%
Milestone 8	19/04	19/04	0	0%
Presentation Session	**20/04/22**	**13/05/22**	**0h**	**0%**
Preparation	20/04	06/05	0	0%
Final Presentation	09/05	13/05	0	0%
Milestone 9	13/05	13/05	0	0%

Figure 25- Full Gantt Chart [20]

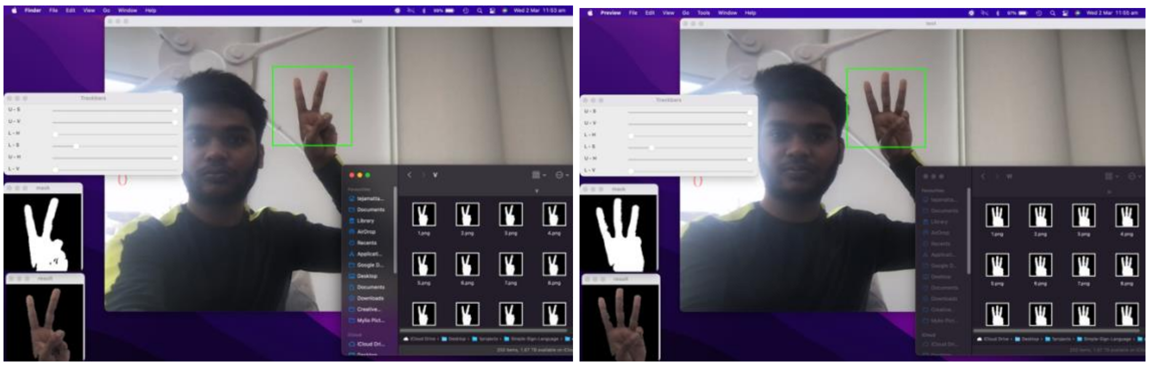
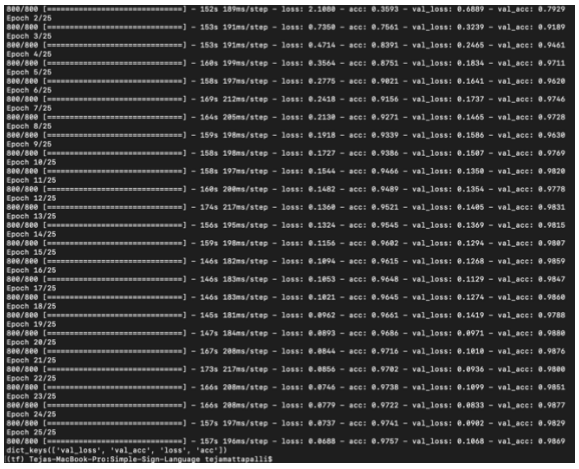
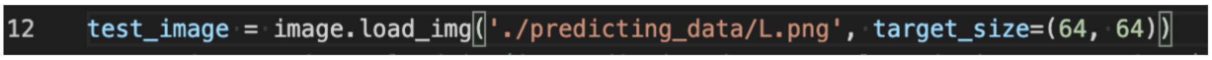


Figure 26 – Dataset Creation



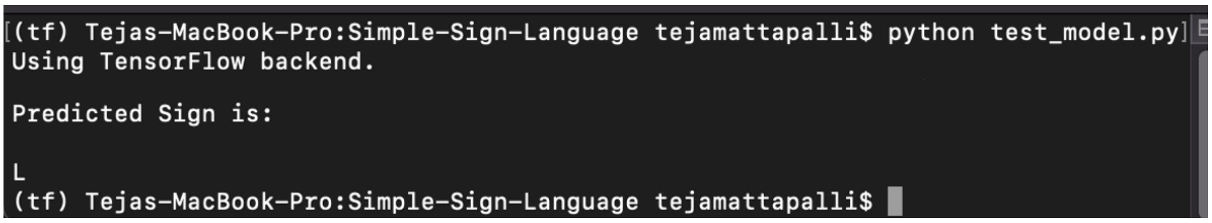
```
800/800 [==============================] - 152s 189ms/step - loss: 2.1080 - acc: 0.3593 - val_loss: 0.6889 - val_acc: 0.7929
Epoch 2/25
800/800 [==============================] - 153s 191ms/step - loss: 0.7350 - acc: 0.7561 - val_loss: 0.3239 - val_acc: 0.9189
Epoch 3/25
800/800 [==============================] - 153s 191ms/step - loss: 0.4714 - acc: 0.8391 - val_loss: 0.2465 - val_acc: 0.9461
Epoch 4/25
800/800 [==============================] - 160s 199ms/step - loss: 0.3564 - acc: 0.8751 - val_loss: 0.1834 - val_acc: 0.9711
Epoch 5/25
800/800 [==============================] - 158s 197ms/step - loss: 0.2775 - acc: 0.9021 - val_loss: 0.1641 - val_acc: 0.9620
Epoch 6/25
800/800 [==============================] - 169s 212ms/step - loss: 0.2418 - acc: 0.9156 - val_loss: 0.1737 - val_acc: 0.9746
Epoch 7/25
800/800 [==============================] - 164s 205ms/step - loss: 0.2130 - acc: 0.9271 - val_loss: 0.1465 - val_acc: 0.9728
Epoch 8/25
800/800 [==============================] - 159s 198ms/step - loss: 0.1918 - acc: 0.9339 - val_loss: 0.1586 - val_acc: 0.9630
Epoch 9/25
800/800 [==============================] - 158s 198ms/step - loss: 0.1727 - acc: 0.9386 - val_loss: 0.1507 - val_acc: 0.9769
Epoch 10/25
800/800 [==============================] - 158s 197ms/step - loss: 0.1544 - acc: 0.9466 - val_loss: 0.1350 - val_acc: 0.9820
Epoch 11/25
800/800 [==============================] - 160s 200ms/step - loss: 0.1482 - acc: 0.9489 - val_loss: 0.1354 - val_acc: 0.9778
Epoch 12/25
800/800 [==============================] - 174s 217ms/step - loss: 0.1360 - acc: 0.9521 - val_loss: 0.1405 - val_acc: 0.9831
Epoch 13/25
800/800 [==============================] - 156s 195ms/step - loss: 0.1324 - acc: 0.9545 - val_loss: 0.1369 - val_acc: 0.9815
Epoch 14/25
800/800 [==============================] - 159s 198ms/step - loss: 0.1156 - acc: 0.9602 - val_loss: 0.1294 - val_acc: 0.9807
Epoch 15/25
800/800 [==============================] - 146s 182ms/step - loss: 0.1094 - acc: 0.9615 - val_loss: 0.1268 - val_acc: 0.9859
Epoch 16/25
800/800 [==============================] - 146s 183ms/step - loss: 0.1053 - acc: 0.9648 - val_loss: 0.1129 - val_acc: 0.9847
Epoch 17/25
800/800 [==============================] - 146s 183ms/step - loss: 0.1021 - acc: 0.9645 - val_loss: 0.1274 - val_acc: 0.9860
Epoch 18/25
800/800 [==============================] - 145s 181ms/step - loss: 0.0962 - acc: 0.9661 - val_loss: 0.1419 - val_acc: 0.9788
Epoch 19/25
800/800 [==============================] - 147s 184ms/step - loss: 0.0893 - acc: 0.9686 - val_loss: 0.0971 - val_acc: 0.9880
Epoch 20/25
800/800 [==============================] - 167s 208ms/step - loss: 0.0844 - acc: 0.9716 - val_loss: 0.1010 - val_acc: 0.9876
Epoch 21/25
800/800 [==============================] - 173s 217ms/step - loss: 0.0856 - acc: 0.9702 - val_loss: 0.0936 - val_acc: 0.9800
Epoch 22/25
800/800 [==============================] - 166s 208ms/step - loss: 0.0746 - acc: 0.9738 - val_loss: 0.1099 - val_acc: 0.9851
Epoch 23/25
800/800 [==============================] - 166s 208ms/step - loss: 0.0779 - acc: 0.9722 - val_loss: 0.0833 - val_acc: 0.9877
Epoch 24/25
800/800 [==============================] - 157s 197ms/step - loss: 0.0737 - acc: 0.9741 - val_loss: 0.0902 - val_acc: 0.9829
Epoch 25/25
800/800 [==============================] - 157s 196ms/step - loss: 0.0688 - acc: 0.9757 - val_loss: 0.1068 - val_acc: 0.9869
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
(tf) Tejas-MacBook-Pro:Simple-Sign-Language tejamattapalli$
```

Figure 27 – Training CNN



```
test_image = image.load_img('./predicting_data/L.png', target_size=(64, 64))
```

Figure 28 – Trained mode



```
(tf) Tejas-MacBook-Pro:Simple-Sign-Language tejamattapalli$ python test_model.py
Using TensorFlow backend.

Predicted Sign is:

L
(tf) Tejas-MacBook-Pro:Simple-Sign-Language tejamattapalli$
```

Figure 29 – Detection Model(upload)

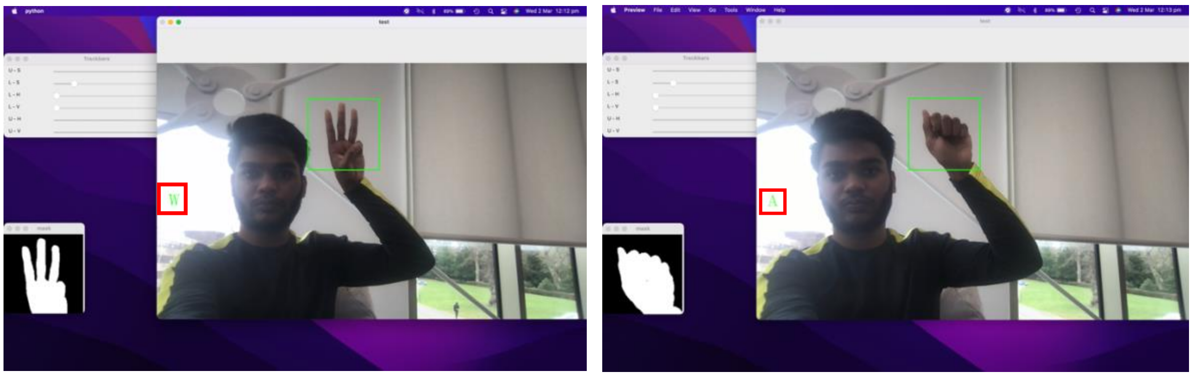
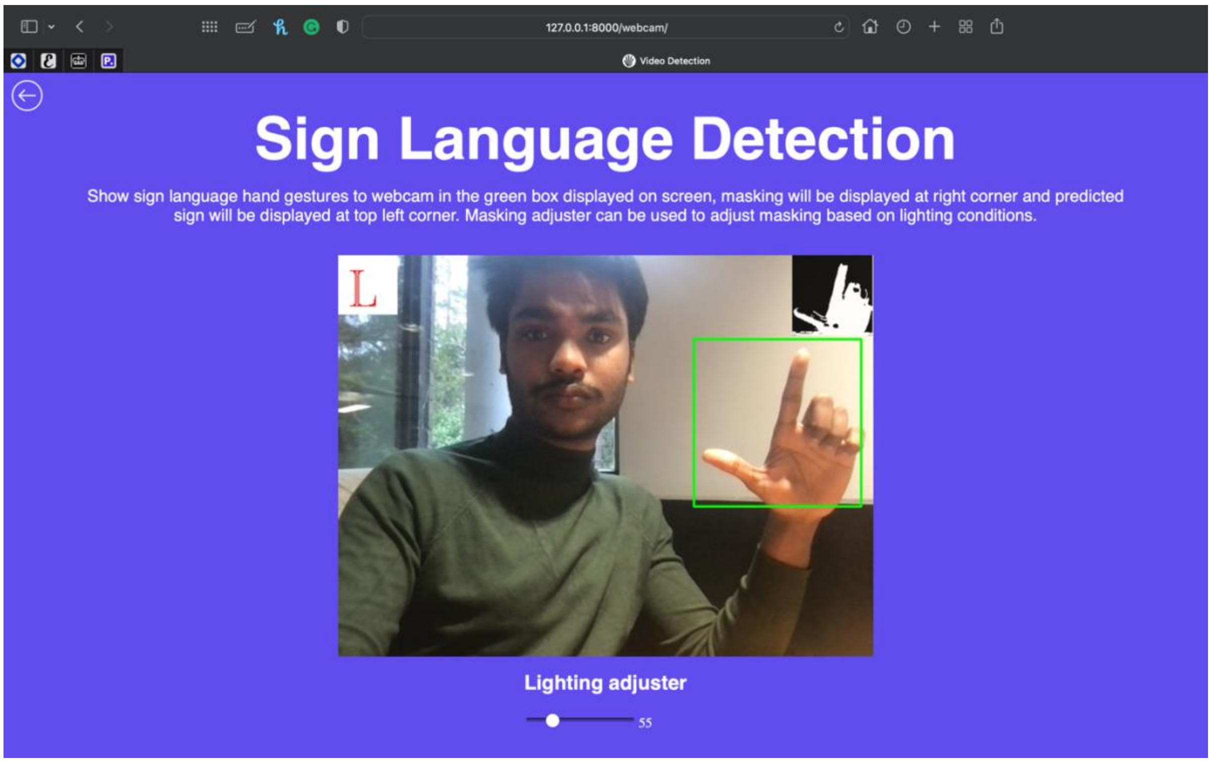


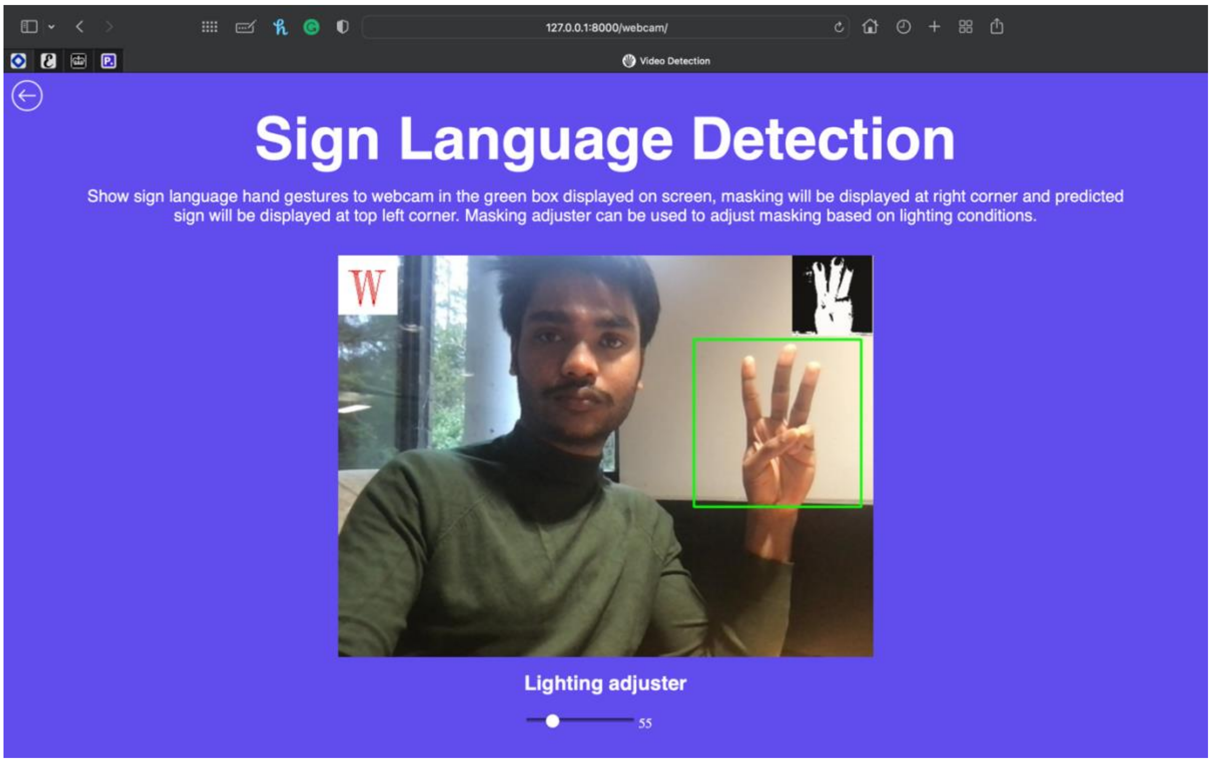
Figure 30– Detection Model (Webcam)



Sign Language Detection

Show sign language hand gestures to webcam in the green box displayed on screen, masking will be displayed at right corner and predicted sign will be displayed at top left corner. Masking adjuster can be used to adjust masking based on lighting conditions.

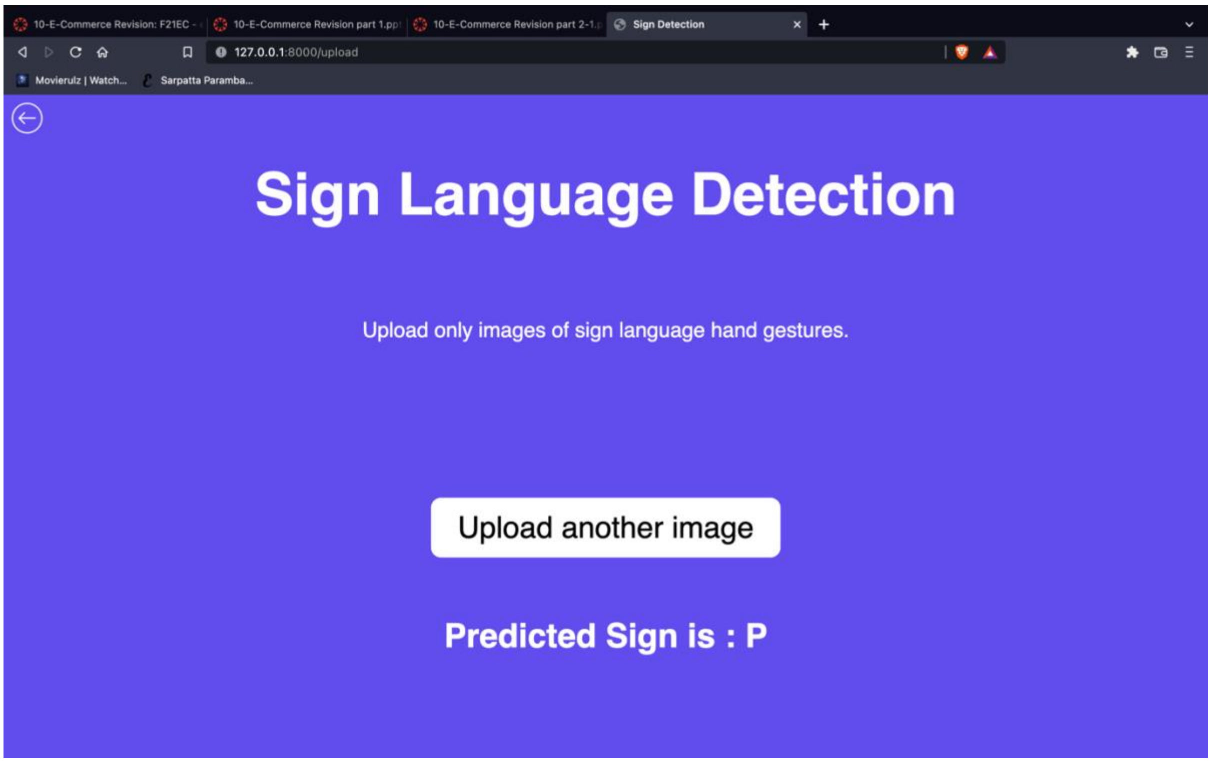
Figure 31 – Webcam Page (Result: L)



Sign Language Detection

Show sign language hand gestures to webcam in the green box displayed on screen, masking will be displayed at right corner and predicted sign will be displayed at top left corner. Masking adjuster can be used to adjust masking based on lighting conditions.

Figure 32 – Webcam Page (Result: W)


10-E-Commerce Revision: F21EC
10-E-Commerce Revision part 1.ppt
10-E-Commerce Revision part 2-1
Sign Detection
127.0.0.1:8000/upload
Movierulz | Watch...
Sarpatta Paramba...

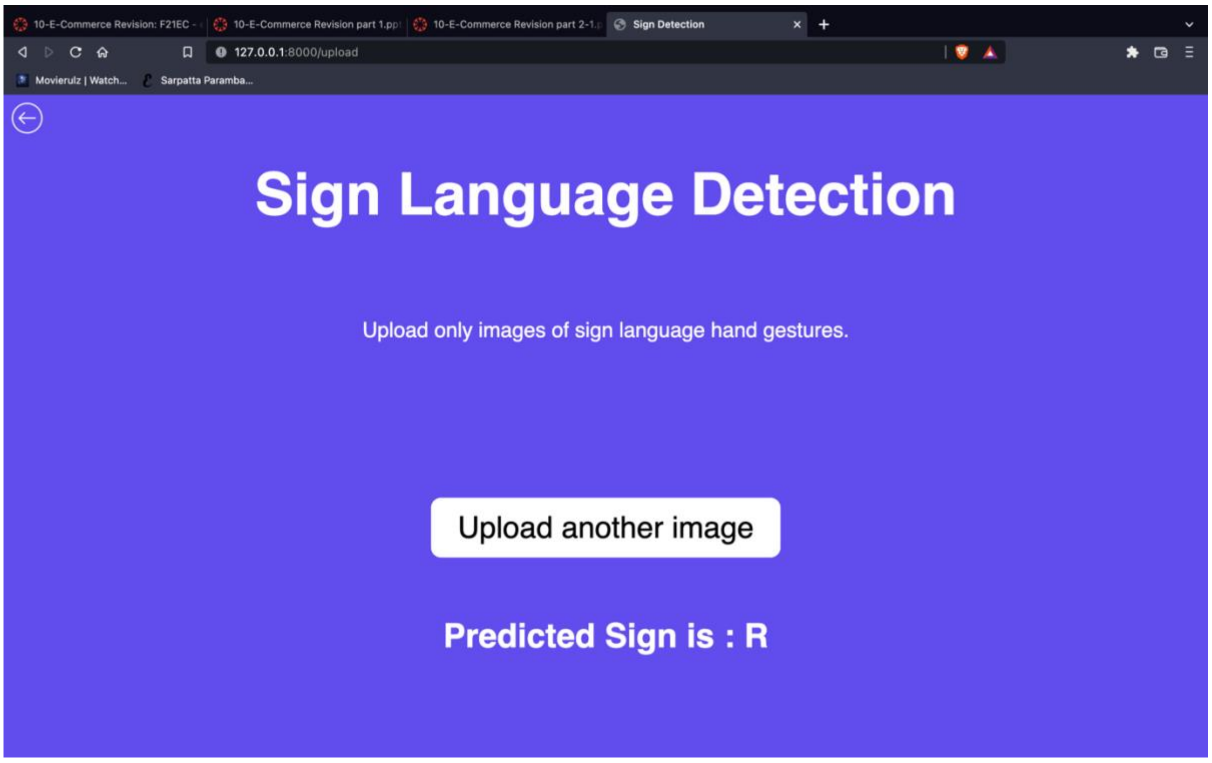
Sign Language Detection

Upload only images of sign language hand gestures.

Upload another image

Predicted Sign is : P

Figure 33 – Upload Page (Result: P)


10-E-Commerce Revision: F21EC
10-E-Commerce Revision part 1.ppt
10-E-Commerce Revision part 2-1
Sign Detection
127.0.0.1:8000/upload
Movierulz | Watch...
Sarpatta Paramba...

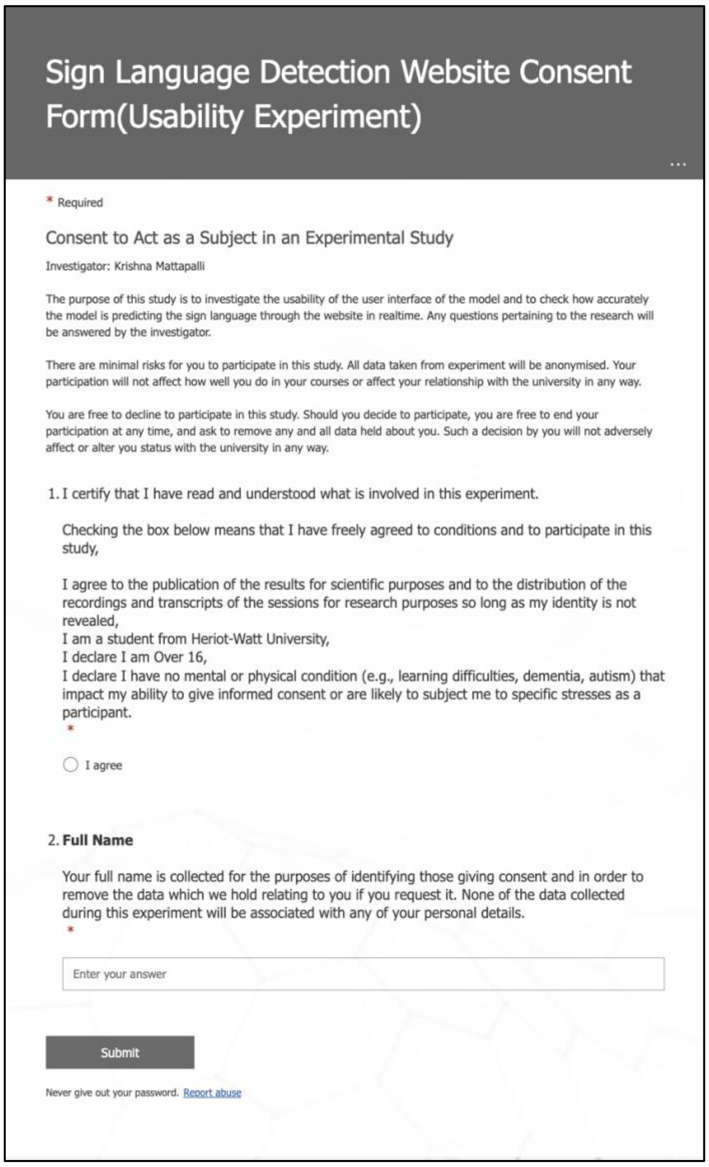
Sign Language Detection

Upload only images of sign language hand gestures.

Upload another image

Predicted Sign is : R

Figure 34 – Upload Page (Result: R)



Sign Language Detection Website Consent Form(Usability Experiment)

* Required

Consent to Act as a Subject in an Experimental Study

Investigator: Krishna Mattapalli

The purpose of this study is to investigate the usability of the user interface of the model and to check how accurately the model is predicting the sign language through the website in realtime. Any questions pertaining to the research will be answered by the investigator.

There are minimal risks for you to participate in this study. All data taken from experiment will be anonymised. Your participation will not affect how well you do in your courses or affect your relationship with the university in any way.

You are free to decline to participate in this study. Should you decide to participate, you are free to end your participation at any time, and ask to remove any and all data held about you. Such a decision by you will not adversely affect or alter you status with the university in any way.

1. I certify that I have read and understood what is involved in this experiment.

   Checking the box below means that I have freely agreed to conditions and to participate in this study,

   I agree to the publication of the results for scientific purposes and to the distribution of the recordings and transcripts of the sessions for research purposes so long as my identity is not revealed,
   I am a student from Heriot-Watt University,
   I declare I am Over 16,
   I declare I have no mental or physical condition (e.g., learning difficulties, dementia, autism) that impact my ability to give informed consent or are likely to subject me to specific stresses as a participant.
   *

   ◯ I agree

2. **Full Name**

   Your full name is collected for the purposes of identifying those giving consent and in order to remove the data which we hold relating to you if you request it. None of the data collected during this experiment will be associated with any of your personal details.
   *

   Enter your answer

Figure 35 – Consent Form