Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

SignSavvy: An Intelligent And Interactive Tutoring System For Sign Language

Author:
Rohan Gupta

Supervisor:
Dr Anandha Gopalan

Second Marker:
Dr Chiraag Lala

June 19, 2023

Abstract

Much research has been conducted into the fields of sign language translation and recognition using technology, but very little has been done to address the current limitations of sign language education, with most existing resources failing to provide learners with interactive feedback that is essential for efficient language learning.

We present SignSavvy, an educational platform for interactive sign language learning, deployed as a user-friendly web app. The developed system uses a standard webcam instead of specialised equipment or sensors, leveraging computer vision and deep learning techniques. We develop a feature-based model to classify static signs, achieving near state-of-the-art classification accuracy (90.3%) with real-time performance (0.22s latency), by introducing the novel use of Google's MediaPipe library for hand landmark localisation. We also develop a lightweight custom dynamic model that analyses spatiotemporal aspects of sign language, achieving an impressive classification accuracy (86.7%) through a similarity-based approach, avoiding the need for deep learning models. Additionally, we present various multi-modal teaching mechanisms to improve signer performance and offer targeted suggestions for improvement.

User evaluations and surveys demonstrate positive user experiences and satisfaction with various aspects of SignSavvy. A comparative study with traditional sign language learning methods shows the statistical significance of incorporating interactive teaching and feedback mechanisms into the learning process. This research project contributes to the advancement of the fields of sign language recognition and learning.

Acknowledgements

I would like to express my sincere gratitude to Dr Anandha Gopalan for his valuable guidance and support throughout the project. His genuine passion for the project and continuous feedback have been instrumental in shaping the direction of my research.

I would also like to extend my appreciation to the users who participated in the trials and surveys, generously sharing their feedback, as well as our ASL expert, who provided invaluable input into the nuances of sign language. Their insights and experiences have played a vital role in refining and validating the project. I am grateful for their willingness to contribute their time to be a part of this research.

Lastly, I want to express my heartfelt thanks to my family and friends for their unwavering support and encouragement, whether it be through late-night bug-fixing sessions or countless discussions about the project. Their belief in me has been a constant source of motivation throughout this journey and this project would not have been possible without them.

Contents

Chapter 1

Introduction

Figure 1.1: Concept logo for SignSavvy that uses the ASL sign for "i"

Sign language holds immense importance as a fundamental means of communication for approximately 70 million individuals globally [1] who experience hearing impairments. It plays a critical role in fostering inclusivity and accessibility within societies, serving as the primary language for the deaf community. Through sign language, individuals can effectively express themselves, actively engage in social interactions, and fully participate in various aspects of life. However, despite its widespread use, sign language continues to encounter substantial challenges in terms of accessibility and education.

While interactive educational platforms like Duolingo [2] have gained immense popularity for spoken languages, they fall short when it comes to sign language. These platforms provide learners with engaging experiences and feedback-driven exercises to gain proficiency in spoken languages. The lack of comprehensive and interactive resources for learning sign language is a major concern [3], considering the substantial and growing number of individuals who rely on sign language for communication [4]. Existing learning resources primarily consist of instructional videos, which, while valuable, fail to provide an effective and immersive learning experience. When learning languages, receiving validation and feedback is crucial to enhance the learning process and simulate the guidance of a real-life tutor.

Sign language poses unique challenges compared to spoken languages. Unlike sequential word-based structures, sign language sentence structures often involve multiple movements performed in parallel. Context and non-manual features, such as facial expressions, mouth shapes and head/torso movements add additional meaning to signs, akin to tone in spoken languages [5]. Factors like hand orientation, wrist angle, and other subtle nuances further complicate the learning process. Learning sign language solely through videos without interactive feedback becomes challenging due to these complexities.

However, to overcome these complexities and create an inclusive learning environment, an intelligent tutoring system (ITS) for sign language is a promising solution. An ITS would not only validate learners' sign performance but also provide personalised feedback and targeted suggestions for improvement, simulating the guidance of a real-life tutor. By leveraging computer vision and deep learning techniques, we can harness the power of technology to develop an ITS that offers a dynamic and interactive learning experience tailored specifically to sign language.

1.1 Objectives

The main objectives of our research can be summarised as follows:

1. **Enabling widespread accessibility of sign language learning:** We aim to design a system that will work seamlessly with commonly available hardware, such as phone cameras or laptop webcams, eliminating the need for specialised equipment or sensors. Furthermore, the platform should be accessible as a user-friendly deployed web app, ensuring widespread availability and ease of use for learners across different devices and platforms.
2. **Encompassing various aspects of sign language:** We aim to design a system that is able to recognise and encompass various aspects of both static signs (signs that involve no movement) and dynamic signs (signs that involve movement).
3. **Personalised feedback-based learning:** We aim to design a system that provides learners with real-time multi-modal feedback to facilitate progress in acquiring sign language proficiency and provide a comprehensive learning experience.

By making learning to sign a more enjoyable and interactive experience, it opens up the possibility for many people to learn such an important and widely-used skill. Our goal is to fill the gap in sign language education and provide a transformative learning experience for people with hearing impairments, as well as those seeking to communicate inclusively or expand their linguistic abilities.

1.2 Contributions

Specifically, the main contributions of this thesis can be summarised as follows:

1. **Exploration and Development of Static Sign Language Recognition Pipelines:**
   - We investigate and evaluate various static sign language recognition pipeline (SLR) architectures to identify state-of-the-art systems and approaches.
   - We develop a high-performing feature-based static model that achieves results comparable to state-of-the-art methods, ensuring reliable recognition of static signs (Chapter 3).
   - Our static pipeline is designed for interpretability and real-time usage, making it suitable for interactive learning environments and providing learners with immediate feedback on their signing performance.
   - We introduce the novel use of the MediaPipe library [6] in the static SLR pipeline, enhancing accuracy and performance by using hand landmark coordinates as features.
2. **Creation of a Custom Dynamic Sign Language Recognition Pipeline**:
   - We develop a custom dynamic sign language recognition pipeline (Chapter 4) to address the challenges associated with recognising and interpreting dynamic signs, which make up a significant portion of sign language vocabulary.
   - Our dynamic pipeline is designed to capture and analyse the spatiotemporal aspects of sign language, improving the overall accuracy and effectiveness of sign recognition.
   - We design the dynamic pipeline to handle variations in signing speed, motion dynamics and timing, making it robust and adaptable to different signing styles and variations.
3. **Construction of SignSavvy Application:**
   - We build SignSavvy, an accessible and user-friendly application that integrates the developed static and dynamic sign language recognition pipelines to provide a comprehensive learning experience for sign language learners (Section 5.1).
   - We develop a user interface that allows users to interactively learn and practice sign language, offering a seamless and immersive learning environment. We design the platform iteratively based on continuous user feedback and human-centred design principles.

4. **Development of Feedback Mechanisms for Sign Language Learners:**
    - We implement multi-modal feedback mechanisms within SignSavvy to deliver accurate and personalised feedback to signers (Section 5.2).
    - We provide targeted suggestions for improvement based on the analysis of learners' sign performance, simulating the guidance of a real-life tutor.
    - We enhance the learning process and sign proficiency by offering comprehensive validation and feedback on sign performance in real-time.
    - We conduct various in-depth user evaluations to assess the effectiveness of the platform and different feedback mechanisms, including a comparative study with traditional ASL learning methods (Chapter 6).

1.3 Ethical Considerations

When developing SignSavvy, there are several ethical considerations which we must consider. These are summarised below:

- **Accessibility:** With sign language primarily being a tool for communication for deaf and hard-of-hearing individuals, it is essential that a tutoring system should be designed around being accessible to these people, and more generally to people with any accessibility issues or disabilities. By being inclusive and usable for all users, the app would avoid any ethical issues around discrimination or marginalisation.
- **Data Privacy:** The app will collect information about the user in the form of recordings through the user's webcam when they attempt signs. Also, the app will store data on users' previous strengths and weaknesses so it can increase the frequency of signs that the individual finds difficult. Whilst there is no plan to store any of this data between runs of the program initially, this is an important consideration for the future. Building a secure application and informing users about what data is being collected is essential. There are also considerations around the recordings contained in training datasets, often containing close-up and personally-identifying recordings of many individuals' likenesses, including at least their face and torso [7].
- **Fairness and Reliability of the Model:** As with any Computer Vision System, it is important that the system should be designed and trained around a wide and diverse dataset so that it is not biased in aspects including race and gender [8]. Inaccurate recognition of signs could lead to frustration and discrimination against certain groups of users.
- **Cultural Sensitivity:** Sign language, like spoken languages, is not a monolithic concept but has variations between regions. Even within Britain and the use of British Sign Language, there exist a multitude of dialects and variations in different regions [9] (for example, London and Newcastle each have their own unique regional signs). When teaching sign language, it is important to recognise that all forms of sign language are equally valid and not impose the version we teach as the only correct way. Furthermore, it should be made clear SignSavvy is not an official application to learn sign language and acts as a proof of concept for the technology rather than a recognised method of certification.

Chapter 2

Background

2.1 Sign Language

Figure 2.1: Signs for the word *"Student"* in American, Italian and Thai Sign Languages [5]

Sign language is a visual form of communication that uses hand gestures, facial expressions and body language to convey meaning. The significance of sign language lies in its ability to provide a means of communication to those with speech or hearing impairments. Sign language is also essential in promoting cultural understanding and acceptance of deaf and hard-of-hearing communities. Language is inseparable from culture and so sign language forms an integral part of the identities of hard-of-hearing people

There is evidence that sign language, in a basic form, has been around for thousands of years [10]. As of now, there are hundreds of sign languages belonging to different regions, each of which is unique and has its own structure, grammar and lexicon. The most common variations are the British Sign Language (BSL), the American Sign Language (ASL) and the Australian Sign Language (Auslan), each of which has its unique characteristics and signs. [11]

According to the WHO [4], over 5% of the world's population (430 million people), have some form of hearing loss, with the number expected to be closer to 700 million people, or one in every ten, by 2050. Furthermore, of the roughly 70 million deaf people, around 80% are illiterate [1]. This demonstrates the significant and growing need for sign language education, both for those with disabilities and the general population to promote an inclusive environment. Additionally, for those without any disabilities themselves, research shows that learning sign language can provide some cognitive benefits, improving visual-spatial skills and problem-solving ability [12].

2.1.1 American Sign Language (ASL)

American Sign Language, abbreviated to *ASL*, is a natural language used by members of the North American Deaf community. We choose to focus on ASL in this thesis as it is the most widely used sign language around the world and there is a huge availability of data and literature.

ASL signs consist of five basic parts [5] (known as parameters), these are:

- Hand Shape - hand configuration that consists of the manual alphabet and other variations
- Hand Orientation - which direction palm is facing e.g. palm facing out, palm up
- Location - relative to rest of body e.g. chin, front of body, shoulder
- Movement - changing location in physical space e.g. in a circle, forward, up and down
- Non-manual signals (facial expressions and head/body positions) e.g. eyebrows, eyes, lips

Different signs can share one or more parameters, making it essential to consider the combination of all the parts. Context and non-manual signals are integral to the semantics of ASL. Furthermore, static signs only make up a small proportion of all ASL signs, with continuous signs required to capture the temporal nature of ASL's dynamic aspects.

2.2 Sign Language Recognition (SLR)

There has been extensive research conducted in the field of sign language recognition (SLR), which is defined as the computational task of recognising actions from sign languages [13]. For a computer-based system to accurately recognise hand gestures, there are a variety of different approaches and algorithms. More robust systems also include non-manual (e.g. facial expression) information and can detect continuous signs through analysing spatial-temporal information, instead of just isolated signs from static images.

Sign language recognition has several challenges in its implementation, some of which are outlined below:

- **Image processing** - There is considerable added complexity when considering aspects like rotation, translation, scale and complex backgrounds [14]. Many approaches have been used to simplify the problem of image segmentation (see Section 2.2.5).
- **Motion Analysis** - Predictability is affected by factors such as background illumination and speed of movement. Furthermore, a difference in viewpoints causes some distortion of gestures in 2D space [15].
- **Consideration of non-manual signs and semantic context** - As mentioned in Section 2.1, ASL consists of more than just hand gestures and context and facial expressions are also important to properly represent the language vocabulary.

2.2.1 Types of SLR and Data Acquisition Devices

There are two key approaches to hand gesture recognition for SLR, *vision-based* and *sensor-based*.

Vision-based approaches involve the acquisition of images and videos through a camera, such as a webcam or a more advanced imaging device like Microsoft's Kinect (which provides both detailed colour and depth information). From these samples, features such as the finger positions, joint angles and palms are extracted and then used to classify the sign.

Sensor-based approaches involve the use of sensors to capture features such as the velocity, movement and relative position of the hand [15]. For sensor-based approaches, data gloves are commonly employed and worn by users. The sensor outputs on the data gloves are converted into electrical signals which can then be used in recognition.

Although numerous studies have achieved a higher accuracy using data gloves [16, 17], and the approach has even been explored in a sign language tutoring context [18], they remain very costly and unnatural for their users. Therefore, this thesis will approach SLR from a vision-based approach.

Similarly, although the Kinect is widely available, it is also a costly and inaccessible option for most. As sign language tutoring emphasises inclusivity and accessibility, we will focus on standard webcam-recorded images and videos in this thesis. Webcams are inexpensive, widely owned and convenient for users. As the quality of data provided by a webcam is relatively low, recognition performance will be significantly more affected by environmental factors such as skin tone, lighting conditions or obstructions. To reduce the effect of these variables, it is essential to employ image enhancement techniques (see Section 2.2.4)

2.2.2 Vision-based SLR Pipeline

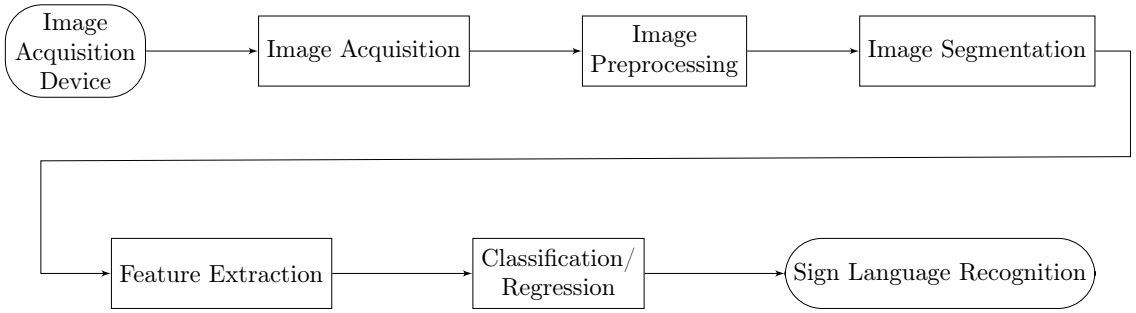


Figure 2.2: Typical pipeline for vision-based sign language recognition

Vision-based SLR consists of five main stages, as shown in Figure 2.2: image acquisition, image preprocessing, image segmentation, feature extraction and finally classification. A brief overview of these stages is summarised as follows:

1. **Image acquisition** (Section 2.2.1 and Section 2.2.3): Images can be acquired manually or from publicly available data sets. These images will later be used in training the model and is it important to obtain data for a wide variety of different signs coming from different signers and in different conditions.
2. **Image preprocessing** (Section 2.2.4): Elimination of unwanted noise and image quality enhancements to prepare it for the later steps.
3. **Image segmentation** (Section 2.2.5): Segmentation and tracking of areas of interest, particularly the separation of the hand from the background in the case of SLR.
4. **Feature extraction** (Section 2.2.6): Identifying and extracting useful information from the segmented images and videos, with the image regions being converted into feature vectors, ready for recognition.
5. **Classification** (Section 2.2.7): A machine learning model will be trained on the database of labelled signs and will form a relationship between extracted features and corresponding labels representing the sign's meaning. The trained model will be able to predict the label of the user-performed sign based on its extracted features in the previous step.

2.2.3 ASL Datasets

There are numerous open-source datasets available for training and evaluating the performance of American sign language models. A suitable dataset will facilitate both training and basic evaluation. Some data sets contain images for static sign language, whilst others contain videos which can be used for continuous sign language applications.

Static/Finger Spelling datasets - One popular static dataset is the *ChicagoFSWild+* dataset [20], which includes over 55,232 annotated images of hand signs for individual letters of the ASL alphabet, signed by 260 individuals. Another notable dataset is the *ASL-LEX* dataset [21], which includes over 60,000 images of various signs from the ASL lexicon, as well as detailed annotations

Figure 2.3: Samples from commonly used sign language datasets [19]

for each image. Similarly, the *Synthetic ASL Alphabet dataset* [22] provides over 27,000 high-quality generated images of the ASL alphabet.

Datasets containing video - For ASL, there exist numerous different popular datasets which include some form of continuous signs and videos, facilitating a much larger vocabulary than the static data sets:

- `Purdue RVL-SLLL` [23] - consists of gestures, movements, words and sentences signed by 14 signers and made up of 2576 videos (184 videos per signer). 39 of the videos are isolated motion primitives and 62 of them consist of hand shapes and sentences.
- `American Sign Language Lexicon Video Dataset (ASLLVD)` [24] - consists of high-quality videos of approximately 3800 ASL signs, signed by 4 signers
- `RWTHBOSTON-104` [25] - consists of isolated sign language samples to form a vocabulary of 104 signs and 201 sentences, signed by 3 signers.
- `RWTHBOSTON-400` [26] - consists of continuous sign language samples to form a vocabulary of 406 words and 843 sentences, signed by 4 signers
- `Word-Level American Sign Language` [27] - consists of over 12000 videos and represents the largest video dataset for ASL, WLASL features 2000 common different words in ASL

2.2.4 Image Preprocessing Techniques

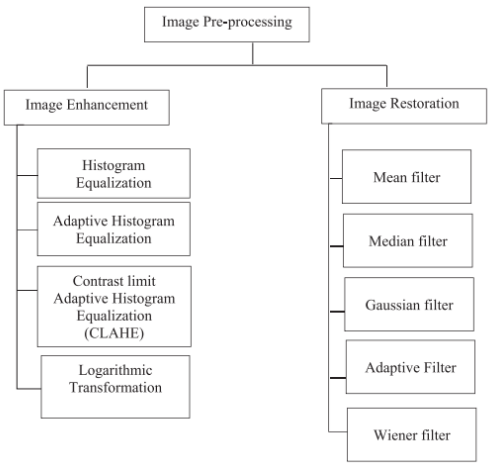


Figure 2.4: A breakdown of popular image preprocessing algorithms [28]

Image preprocessing is the first stage of the SLR pipeline, and converts the raw images and videos acquired from the dataset or camera and removes unwanted noise and enhances image quality. Numerous algorithms exist to transform images (summarised in Figure 2.4), such that the overall accuracy of the detection is much more accurate. The algorithms can be classified into two broad groups, **image enhancement** and **image restoration** [28].

Image enhancement is the process of restoring an image's visual appearance and improving quality for analysis further down the pipeline. Some common techniques are described below:

- *Histogram Equalisation (HE)*: A method employed to enhance the colour and contrast of an image by modifying the intensity values of its pixels [29]. By redistributing the most common intensity values across the entire intensity range, this technique improves the sharpness of edges and boundaries in images while reducing local details.

$$pdf(A_k) = \frac{n_k}{n} \quad where\ k=0,\ 1,\ 2,\ ...,\ L\text{-}1$$

$$cdf(A_k) = \sum_{j=0}^{k} P(A_j) = \sum_{j=0}^{k} \frac{n_k}{n}$$

  The process of histogram equalisation begins by computing the histogram of the image (a graph showing the number of pixels with each intensity value) and the cumulative distribution function (CDF) of the histogram. The CDF is then used to map the intensity values of the pixel in the original image to new intensity values, thereby equalising the histogram. The equations above are used to calculate the probability density function and the cumulative distribution function respectively [29], with k denoting the range of the intensity value and n_k being the number of pixels with intensity value of A_k within the image A.

  The technique has been used in various SLR studies [30] to improve the brightness and contrast of input images. The main advantage of histogram equalisation is that it is simple and very effective for grayscale images, particularly when the intensity values are concentrated in a narrow range. However, it can also introduce background noise and can make the image appear unnatural or unrealistic. Furthermore, the technique can lead to over-enhancement and loss of detail in the image.

- *Adaptive Histogram Equalisation (AHE)*: A variant to the standard HE algorithm described above, AHE [31] is applied locally to an image. Instead of processing the entire image, AHE divides the image into small blocks, called *"tiles"*, and applies HE to each tile independently. This allows AHE to preserve local features and avoid over-saturating the image, meaning it outperforms the standard HE technique. However, AHE still has a negative effect on the output as it tends to amplify noise and fails to retain the original brightness of an image.

Image restoration is the process of restoring an image that has been degraded due to noise and blurring. We can use a wide variety of algorithms to remove noise, depending on the type and quantity of noise present in the image. Some common filters are described below:

- *Mean Filter*: A mean filter smooths an image by replacing each pixel's value with the arithmetic average of the pixel in its surrounding neighbourhood. The neighbourhood is defined by a sliding window with the size of the window called the filter's *kernel size*.

  Despite being simple and easy to implement, it is not ideal for all types of noise and can cause blurring of edges and loss of image details. As per comparisons on restoration techniques used in SLR [15], mean filtering is not often used in studies, and median and Gaussian filters are preferred.

- *Median Filter*: A median filter follows a similar sliding window approach to the mean filter, but instead replaces a pixel's value with the median pixel value of the neighbourhood pixels. This is calculated by sorting the neighbourhood pixel values in numerical order and using the middle value (or the average of the two middle values if there is an even number of pixels).

The median filter is particularly effective at removing "salt and pepper" noise [28], which are pixels that are either much darker or much brighter than the surrounding pixels. The median filter is also robust to outliers and can be less aggressive than the mean filter (preserving the edges and details of an image), however it is less effective at removing Gaussian noise.

- *Gaussian Filter*: Unlike the mean filter, the Gaussian filter assigns different weights to each pixel depending on its distance to the central pixel. The weights are calculated using a Gaussian function, which gives a higher weight to pixels closer to the centre.

  In practice, we implement a Gaussian filter by convolving the image with a kernel that is a 2D Gaussian function formed by calculating the value of the Gaussian function for each pixel within a certain radius. The standard deviation (σ) represents the radius and determines the "blurriness" of the filter or the amount of soothing that will be applied to the input image. The equation for a two-dimensional Gaussian filter is given by:

  $$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \textit{where } x \textit{ and } y \textit{ denote row and column values}$$

  The Gaussian filter is the most widely used in image processing, including in several SLR papers. It offers effective removal of noise, whilst preserving details and edges of an image. It is also a good choice for blurring and smoothing images. However, the Gaussian filter is not as effective as the median filter for salt and pepper noise and has a relatively high computation time.

- *Adaptive Filter*: An adaptive filter is one that can adjust its parameters based on the characteristics of the input signal. The filter removes noise whilst preserving detail, meaning it preserves edges better than other filters. A commonly used adaptive filter is known as the *Wiener filter*, which is not sensitive to noise and exploits the statistical properties to determine the filter coefficients [28]. The algorithm minimises the mean square error between the restored and the original image and is particularly effective on images that have been affected by white Gaussian noise.

2.2.5 Image Segmentation and Tracking Techniques

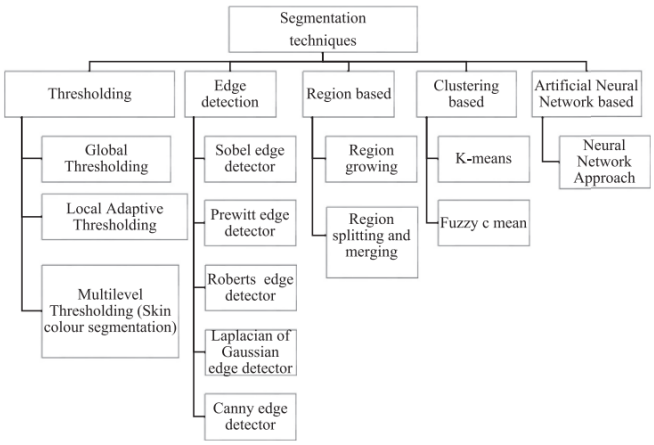


Figure 2.5: A breakdown of popular image segmentation algorithms [28]

Image segmentation is the process of partitioning images into multiple regions [32] [28], with each corresponding to a distinct object or part of the image. In SLR, segmenting the hands from the background and identifying other important regions of interest is essential. Image segmentation can be either contextual, using spatial relationships between features (e.g. edge detection) or non-contextual, where pixels are grouped based on global attributes (e.g. thresholds).

There are various techniques used in image segmentation. One popular approach is based on thresholding, in which pixels in an image are classified depending on whether their value is greater than or

less than some threshold value. The threshold can be global and take one value for the whole image, or be locally adaptive and have different values for each sub-image of the original image. A Multi-level threshold has been used for skin colour segmentation in numerous studies [33], a crucial task for separating a hand from its background in SLR. The approach determines multiple thresholds and divides the image into distinct regions (rather than just two classes like bi-level thresholding). The colour models for multilevel thresholding include the RGB (Red/Green/Blue), HSV (Hue Saturation Value), Y (Luminance) and YCbCr (Luminance+Chromaticity) colour schemes. HSV and YCbCr are the most extensively used skin colour segmentation techniques in SLR [28].

Edge detection is another classic image processing technique. The technique is based on quick changes in intensity values and is used to identify, eliminate and join edges. Many edge detection algorithms determine edges by comparing the first derivative of intensity to a threshold or by examining zero crossings in the second derivative. Popular edge detectors include the Robert, Sobel, Prewitt, Laplacian of Gaussian and Canny edge detectors. The Canny edge detector is a popular choice because of its ability to detect edges of varying widths. It is able to detect edges with a high level of accuracy and suppress noise, and even works well in varying light conditions. The algorithm for Canny edge detection [33] is a multi-stage process as described below:

1. Smooth the image with a Gaussian filter to remove noise (see Section 2.2.4)
2. Apply a Sobel filter in both dimensions to calculate horizontal gradients (G_x) and vertical gradients (G_y). To apply a Sobel filter, we must convolve the input with a 3 x 3 kernel.
3. Calculate the magnitude of the gradient with the equation $|G| = \sqrt{G_x^2 + G_y^2}$
4. Calculate the edge gradient using the equation $\theta = \arctan\left(\frac{G_x}{G_y}\right)$. The edge directions are then categorised into directions relative to the edge: horizontal, positive, vertical, or negative
5. Apply non-max suppression to remove pixels that do not make up an edge. Check every pixel in the direction of the edge and discard any pixels that are not local maximums in their neighbourhood by setting their value to 0.
6. Perform hysteresis thresholding. A pixel is made strong if the 8 pixels around it are strong, otherwise, it is set to 0. This step will get rid of streaking.

Recently, there has been an increasing amount of skin colour segmentation techniques based on artificial and convolutional neural networks (ANNs and CNNs). Using CNNs, skin regions can be segmented in an image [34] based on features that are learned from training a model with a large amount of labelled data.

Tracking hands in non-static SLR is not a trivial task. Often in SLR, hands will move very quickly and their appearance will change due to changes in orientations and occlusions. In a previous study [35], an algorithm called CamShift was combined with HMMs (see Section 2.2.7) to achieve an end-to-end SLR system for continuous signing.

2.2.6 Feature Extraction

The feature extraction phase is the process of extracting meaningful and interesting information from the input image. The goal of feature extraction is to find a set of distinctive features to represent the image, whilst performing a dimensional reduction from raw pixel data to only the most necessary features. The feature extraction stage takes in an edge map and outputs a feature set, which then is used in the classification stage to identify a specific sign. Features extracted from an image can be summarised into categories based on colour, shape and texture. Examples of geometric (shape) features in SLR include fingertips, finger directions, wrist angles and hand contours. In the creation of the feedback mechanism for Signer, having clearly defined and observable features will provide us with the information required to correct users. There are many algorithms used in feature extraction, popular ones include PCA, SIFT, SURF and Fourier descriptors. Some of these approaches are summarised below:

- **Scale Invariant Feature Transform (SIFT)**: SIFT [36] is a feature extraction technique that is invariant to scale and rotation. SIFT detects and describes scale-invariant interest points, which are robust to changes in scale, rotation, and viewpoint, making them suitable for object recognition tasks like SLR. The algorithm can be broken down into four main steps:

  1. **Determining the approximate location and scale of extrema/keypoints**: The algorithm begins by constructing a scale-space representation of the image, which is a smooth and down-sampled version of the original. $L(x, y, \sigma)$, the scale-space function, is defined as a convolution between the input image I and the Gaussian function, $G(x, y, \sigma)$, giving the equation $L(x, y, \sigma) = G(x, y, \sigma) \cdot I(x, y)$ [28]. The Gaussian function is defined as $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})}$.

     From the scale-space image, the algorithm then identifies extrema (local maxima and minima), which correspond to potential interest points. The Difference-of-Gaussian (DoG) function, $D(x, y, \sigma)$ can be computed by convolving the image with two Gaussian kernels of different standard deviation (σ) differing by a constant scale factor k and then subtracting the smaller scale result from the larger one. Therefore, we obtain the equation

     $$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \cdot I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

  2. **Keypoint localisation**: Keypoints with low contrast or ones that are poorly localised are eliminated.
  3. **Orientation Assignment**: An orientation is assigned to each keypoint based on the image gradient at that keypoint location. This allows for the rotation-invariant matching of the keypoints when compared in different images.
  4. **Feature Descriptor**: Keypoints are transformed into a feature descriptor that describes the local image structure around the keypoint (created through histogramming the neighbourhood region). The descriptor allows for illumination and shape changes.

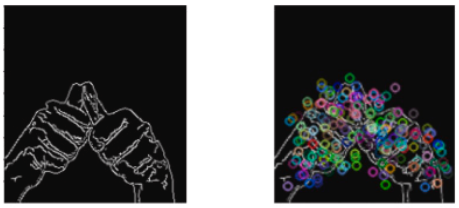


Figure 2.6: Extracted hand features using the SURF algorithm in a study on Indian SLR [37]

- **Speed Up Robust Features (SURF)**: SURF [38] is newer algorithm that is developed based on SIFT. It uses approximate scale-space extrema detection to make computation faster. SURF also uses a smaller 64-dimensional descriptor to describe features (compared to SIFT's 128-dimensional descriptors), further increasing efficiency. SURF is much faster than SIFT and therefore better suited to real-time applications like SLR.

2.2.7 Classification Algorithms used in Static SLR

The final stage of the SLR pipeline involves using a predictor algorithm to create meaningful representations from the features extracted from an image or video, classifying it as a particular sign. Classification algorithms can be either supervised or unsupervised. In supervised machine learning, we train a model on a labelled dataset, where the target variable is already known. The model then recognises the patterns of the input data and is able to make predictions on new, unseen data. Unsupervised machine learning involves training a model on a dataset without labels, where the output variables are unknown. The algorithms attempt to discover patterns or relationships in

the data by grouping together similar observations (using methods like clustering). As numerous labelled datasets are available for ASL (see Section 2.2.3) and signs are categorisable, supervised learning is better suited for most SLR applications. According to a comprehensive literature review of 649 papers in SLR [28], the most common predictors used in the domain are k-nearest-neighbour (KNN), artificial neural network (ANN), support vector machine (SVM), hidden Markov Model (HMM), convolutional neural network (CNN), fuzzy logic and ensemble learning.

- **K-Nearest-Neighbour (KNN):** KNN is a very simple algorithm based on the idea that uses the k number of "nearest" training examples in the feature space to predict the output based on the majority class or an average of the k nearest neighbours. There are numerous ways of measuring distance, such as Euclidean, Manhattan and Mahalanobis distance. KNN has been applied in SLR studies, such as in a study on Indian SLR [39], where a 97.1% accuracy was obtained on numerical ISL sign detection. KNNs have a few limitations, such as being overly sensitive to features that might be irrelevant and not scaling too well to larger datasets or higher dimensions.

- **Convolutional Neural Networks (CNN):** CNNs are a type of artificial neural network that is inspired by the ways animals process visual information. CNN-based methods have emerged as a powerful tool in the computer vision field, differing from traditional fully connected networks by leveraging the power of the hierarchical structure of visual data. CNNs are a type of deep learning algorithm, meaning that features are identified by the network itself rather than by developers.

  CNNs and deep learning approaches are becoming increasingly popular in SLR and other domains. They provide automation of the feature extraction process and can learn to recognise the features of hand gestures such as hand shape and finger positions automatically. When CNNs are trained on sufficiently large amounts of data, they are relatively robust to variabilities such as light conditions, backgrounds and camera angles, making them well-suited to real-time classification from low-resolution inputs like webcams. SLR solutions with CNNs have achieved high accuracy, such as one [40] that used YCbCr for skin colour segmentation and CNNs for feature extraction and classification to obtain a promising accuracy of 98.05% on real-time test data. Model training times and performance can also be greatly improved using methods like transfer learning [41], which is a technique that leverages pre-trained CNNs with popular architectures trained on massive datasets such as ImageNet [42].

2.2.8 Classification Algorithms used in Dynamic SLR

Dynamic SLR pipelines typically require specialised classifiers that can effectively encode and analyse continuous sequences of images or temporal data. There are various predictor algorithms that can be used in dynamic SLR to capture the temporal characteristics of movement whilst performing a sign.

- **Hidden Markov Models (HMM):** HMMs have become the most popular and effective methods of modelling sequences in dynamic SLR. A HMM is a statistical model which is based on a Markov process, with an internal state that is directly observable. The model operates on a hidden set of unknown parameters and can only observe the output state. The observation parameters can then be used to derive their related hidden parameters.

  There are multiple variations of HMM used in studies, including continuous, discrete, parametric and parallel versions. HMMs have also been used in combination with other classification methods like CNNs to create hybrid models [15]. In SLR studies involving using images in video sequences, HMMs have been found to achieve better accuracy than other models [28]. For example, a study [43] used parallel HMMs to classify 22 ASL signs with a recognition accuracy of 94.23% on training data.

  One drawback of using HMMs is the large number of unstructured parameters, which require sizeable datasets to properly train. Additionally, sign language gestures can vary significantly between users, including in motion speed and hand shape, which can be difficult for HMMs to model.

- **Dynamic Time Warping (DTW):** The DTW algorithm [44] is a technique to find optimal alignment between two given, time-dependent sequences. By warping the sequences, DTW allows for comparisons of patterns, even when the temporal sequences are misaligned. The algorithm has been employed in many fields, including speech recognition, financial markets and data mining. More recently, it has also been applied to dynamic SLR, using hand trajectory information to compare a query sign with those in a database of examples. The technique is particularly effective as it accounts for the variation in timings, speeds and duration lengths when performing dynamic signs. One such study [45] applied DTW to HoG features of the signer's pose, achieving an 82% accuracy in ranking signs in the top 10 similarities out of a dataset consisting of 449 dynamic signs.

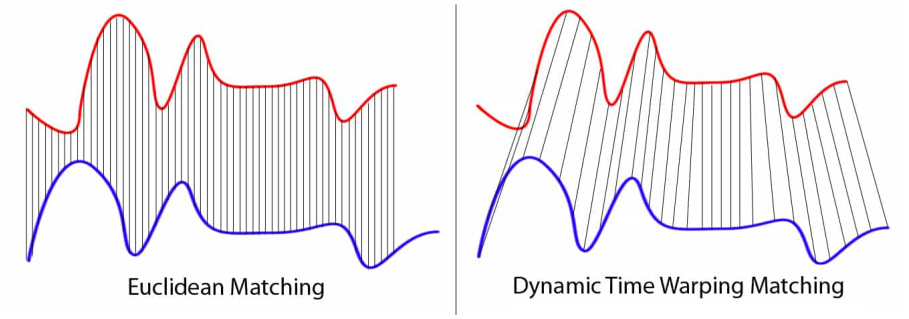


Figure 2.7: A comparison of pattern matching approaches applied to out-of-sync sequences [46]

Given two sequences, $X = [x_1, x_2, \ldots, x_N]$ and $Y = [y_1, y_2, \ldots, y_M]$, where N and M represent the lengths of the sequences, DTW calculates a distance matrix D of size $(N+1)\times(M+1)$, where $D(i, j)$ represents the minimum cumulative distance between the subsequence $X[1 : i]$ and $Y[1 : j]$. The distance $C(i, j)$ between elements $X[i]$ and $Y[j]$ can be computed using a distance metric, such as Euclidean distance or cosine similarity:

$$C(i, j) = d(X[i], Y[j])$$

The algorithm initialises the distance matrix D as follows:

$$D(0, 0) = 0, \quad D(i, j) = \infty \quad \text{for } i = 1 \text{ to } N, \quad j = 1 \text{ to } M$$

It then computes the optimal path by iteratively filling the distance matrix by applying dynamic programming:

$$D(i, j) = C(i, j) + \min(D(i - 1, j), D(i, j - 1), D(i - 1, j - 1))$$

The optimal path represents the alignment between two sequences. It is obtained by backtracking from the bottom-right cell of the distance matrix $D(N, M)$ to the top-left cell $D(0, 0)$ while selecting the neighbouring cell with the minimum cumulative distance at each step. The resulting optimal path P is a sequence of indices that corresponds to the aligned elements of the two sequences. Finally, the DTW distance between the two sequences is calculated as the minimum cumulative distance divided by the length of the optimal path:

$$\text{DTW Distance} = \frac{D(N, M)}{\text{length}(P)}$$

2.3 Tools for Computer Vision and Machine Learning

This section provides a brief introduction to some of the tools and libraries we can use to simplify the computer vision and machine learning algorithms covered in Section 2.2.

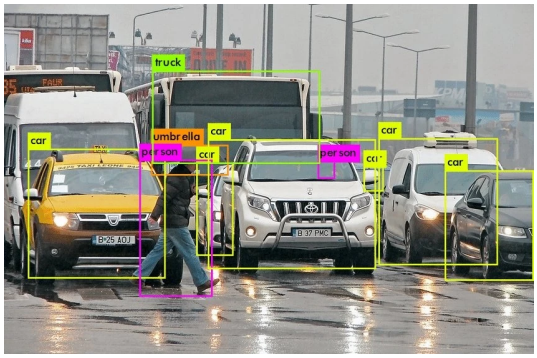


Figure 2.8: Real time object detection using YOLO [47]

2.3.1 YOLO and Single-Shot Detector (SSD) Models

YOLO (You Only Look Once) [48] is a computer vision tool developed by *Ultralytics*, designed specifically for efficient real-time object detection. `YOLOv5` was released in 2020 and is usable in four main versions: small (s), medium (m), large (l) and extra large (xl), with respectively increasing accuracy rates and training times. In 2022, YOLOv7 was released, providing increased accuracy rates, but a slower training time on typical GPUs. There are numerous reasons to use YOLOv5, including the ease of installing the library and fast training speeds.

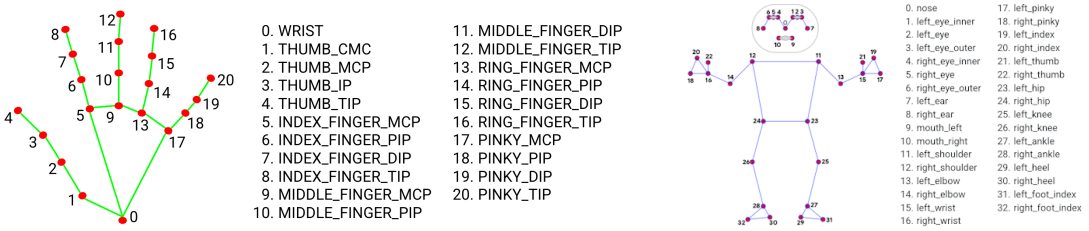
YOLO is a popular example of a single-shot detector (SSD) model. YOLO first divides an input image into a grid and then predicts bounding boxes and class probabilities for each grid cell. YOLO can perform object detection with higher speed and accuracy compared to other object detection algorithms like Faster R-CNN [49] as they avoid the need for bounding box region proposals, instead directly predicting bounding boxes in one pass. An SSD architecture typically consists of a backbone CNN that extracts features from the input image. The features are passed through a series of convolution layers, which capture spatial information at different scales and are then used to predict bounding box coordinates and probabilities. Non-maximal noise suppression is then used to remove redundant overlapping detections and ensure each object is represented by a single bounding box corresponding with the highest confidence score.

2.3.2 MediaPipe Solutions

MediaPipe [6] is an open-source framework developed by Google that allows developers to build and deploy machine learning pipelines for processing multimedia content. It provides a collection of pre-built, reusable, and customisable components for tasks such as object detection, facial landmarks, hand tracking, and gesture recognition. MediaPipe can be used to build solutions that can process live video streams and perform real-time analysis of the content. This can be used to create applications such as augmented reality, real-time object detection, and gesture recognition. It is designed to be highly modular, allowing developers to pick and choose the components that best fit their needs and easily add custom components to the pipeline. MediaPipe also provides interfaces for multiple platforms, including mobile and web, making it easy to deploy pipelines to a wide range of devices. The framework is built on top of OpenCV [50], an open-source computer vision and machine learning library that includes over 2500 optimised algorithms, as well as many other computer vision and graphics libraries. In terms of SLR, two key packages that MediaPipe offers are its hand and pose landmark detection models:

Hand Landmarker Model: The hand landmarker task [51] can be used to localise the keypoints of the hands, as well as render visual effects over the hands. The model can be configured to run on static data or a continuous stream and outputs hand landmarks in 3d image coordinates, as well as handedness (whether a hand is a left or right hand) of multiple detected hands.

The model bundle consists of a palm detection model which locates hands within an input image, and a hand landmarks detection model to identify specific hand landmarks on the cropped image defined by the palm detection model. The detector model is based on a CNN with a single-shot



(a) Hand Landmark Representation [51] **(b)** Pose Landmark Representation [52]

Figure 2.9: MediaPipe Hands and Pose landmark representation mapping

detector architecture, as detailed in Section 2.3.1, taking an RGB frame of video or an image as input. It outputs a float tensor of size 2016 x 18, consisting of predicted embeddings representing anchors which are used in the non-maximum suppression algorithm. The tracker model is based on a CNN with a regression architecture and outputs the keypoint localisation of 21 hand-knuckle coordinates as shown in Figure 2.9a. The model was trained on over 30,000 real-world images, as well as numerous synthetic hand model renders over various backgrounds.

The BlazePalm palm detector operates on palms instead of hands as estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. Similarly to YOLO, the detector model creates 11,136 anchors and then applies non-maximal noise suppression to the output of the SSD model to get a single bounding box for each hand. The architecture of the detector model (shown in Figure 2.10 is based on an encoder-decoder feature extractor, to improve scene-context awareness for different scale objects.

MediaPipe's custom hand landmarker model has three outputs sharing a feature extractor. The architecture, shown in Figure 2.10, shows each head, trained by correspondent datasets marked in the same colour.

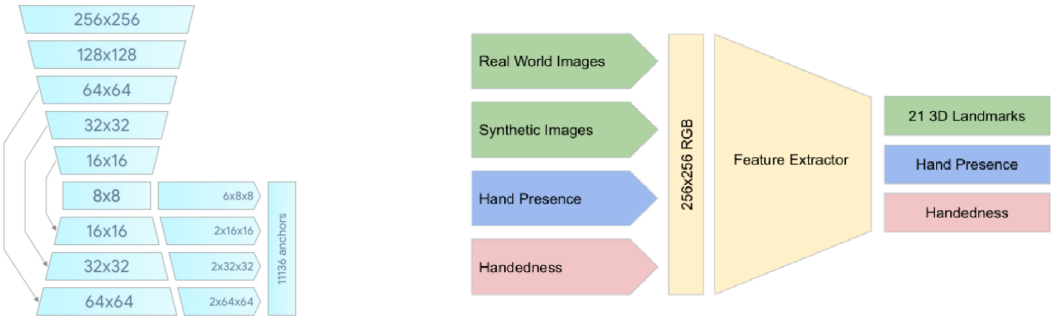


Figure 2.10: MediaPipe Hands architecture: Palm Detector (left) and Hand Landmarker (right)

Pose Landmarker Model: The pose landmarker task [52] uses a model to detect the presence of human bodies within an image frame, and a second model to locate landmarks on the bodies. The pose landmarker outputs 33 3d pose landmarks as shown in Figure 2.9b. The model uses a convolutional neural network with an architecture based on MobileNetV2 [53] and is optimised for on-device, real-time applications. The MobileNetV2 The model also uses GHUM [54], a 3D human shape modelling pipeline, to estimate the full 3D body pose of an individual and predict Z coordinates from 2d images.

The MediaPipe Solutions suite is still currently in alpha stage, but further stable releases are expected in the near future. The tool represents a promising solution for SLR due to its cross-platform and lightweight design. Furthermore, as the models are trained on a large dataset of hand and pose images, the models are relatively robust against factors such as lighting, skin colour, hand pose and occlusions. MediaPipe offers solutions in Python, Android and Web, making it possible to integrate into a browser-based application.

Chapter 3

Static SLR Model Implementation

In this chapter, we present the implementation details of the static sign language recognition model used to classify and analyse still images of ASL signs. Building upon the foundations of SLR mentioned in Section 2.2, we describe our customised approach and techniques that have been tailored to meet the specific objectives of SignSavvy. Throughout this chapter, we delve into various components of our implementation, including our feature extraction, data preprocessing, and classification models.

3.1 Objectives and Proposed Methods

In this section, we clearly define the motivations behind our implementation of a static SLR pipeline, as well as the specific requirements that our designed system should tailor to. Through this, we establish a framework for our subsequent exploration into potential solutions:

1. Development of a static SLR pipeline capable of classifying a broad and non-trivial range of static ASL gestures, including differentiating between signs with small variations.
2. Development of a robust system that is capable of operating on low-resolution images and dealing with a range of real-world challenges, such as variations in light conditions, skin colour, backgrounds, camera angles and occlusions.
3. Development of a system that can be adapted to perform near-real-time classification of signs from a continuous live stream.
4. Investigation and comparison of feature extraction techniques that can accurately model the visual information.
5. Exploration of various data augmentation and preprocessing techniques and their effects on the performance of the SLR model.
6. Evaluation of various different static SLR classification models and architectures to identify the most suitable approach.
7. Evaluation of the robustness, efficiency and scalability of the developed pipeline, considering metrics such as classification accuracy, inference time and resource utilisation.

In order to meet the objectives above, we have chosen to explore the use of convolutional neural networks (see Section 2.2.7) as the primary classification model in the static SLR pipeline. The huge popularity of CNNs in image classification tasks makes them well suited to static signs, with CNNs having been shown to be robust to various visual challenges faced in computer vision when trained on an adequately large and diverse dataset.

3.2 Data Selection and Composition

To train, tune and evaluate our static SLR pipeline, a carefully curated dataset is important to ensure the model is accurate and robust to variations. In this section, we discuss the rationale behind the selection of our dataset, as well as note some of the limitations of our choices.

3.2.1 Static Sign Selection

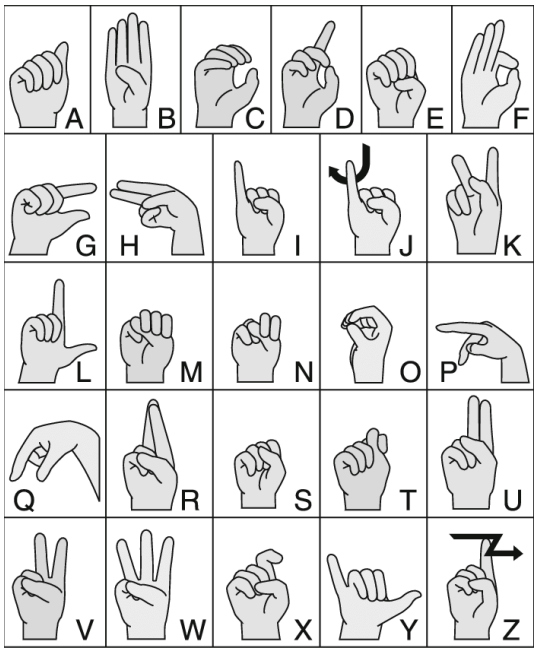


Figure 3.1: The letters of the ASL finger-spelling alphabet [55]

We have chosen to narrow down our 26 static ASL signs, representing the letters of the English alphabet as shown in Figure 3.1. For the letters "J" and "Z", which require motion to sign, we have instead represented them as a static equivalent. The rationale for our decision to use the ASL Alphabet is as follows:

- **Focus on a well-defined set of signs:** Given the large number of words within the SLR domain, it is crucial to focus on a concise set of signs to ensure efficient model training, evaluation, and comparison. By reducing the number of classes, we can maintain better control over the quality and annotation of the data, resulting in more reliable training and evaluation.
- **Availability of Data:** There are numerous high-quality datasets available for the ASL alphabet, with thousands of annotated images. Large amounts of training data are required for good classification performance from a CNN.
- **Importance of fingerspelling in ASL:** The representation of ASL words from individually signed letters, also known as fingerspelling [56], is an essential component of ASL communication. Fingerspelling is used in ASL to spell out names of people and places for which there is not a sign or to represent words that either the signer or the person reading the sign does not know. Fingerspelling signs are also often incorporated into other ASL signs.

3.2.2 Lexset Synthetic ASL Alphabet Dataset

The Synthetic ASL Dataset [22], created by Lexset, contains 27,000 synthetically generated images of the ASL alphabet in front of various backgrounds. It contains a total of 27 folders, one for each letter of the alphabet and one containing a folder of random backgrounds and no hands. Within

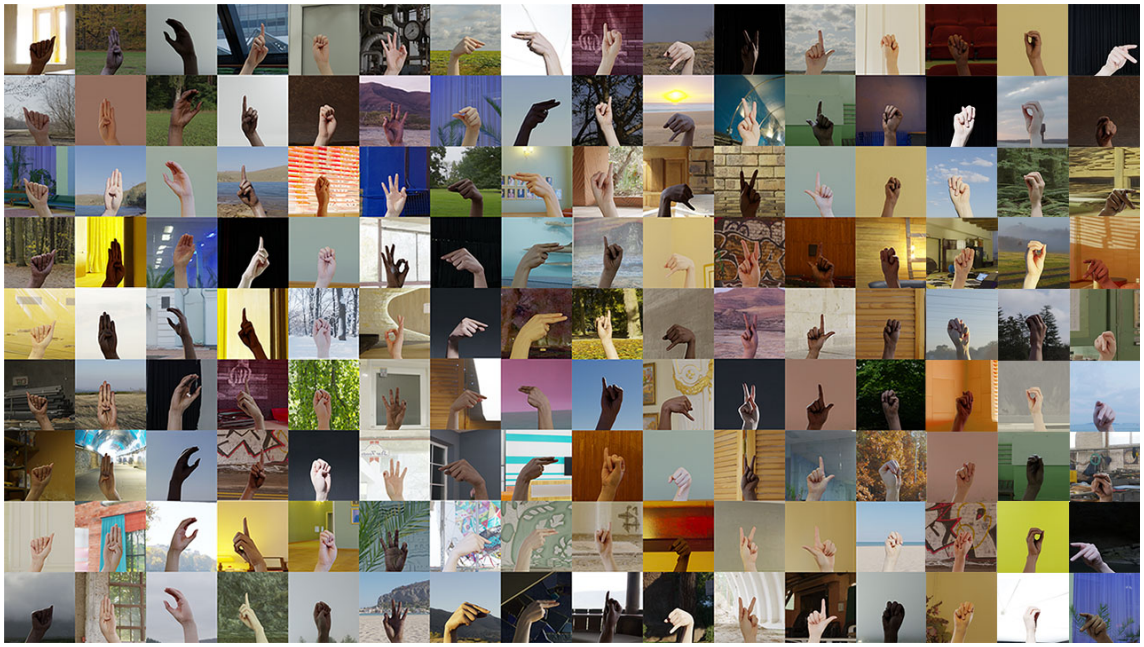


Figure 3.2: Sample of the ASL letters A-P from the Lexset ASL Alphabet Dataset [22]

each folder, there are 900 images intended for training and validating a model and a further 100 images for testing. We choose to remove the random background folder and utilise the remaining 26,000 images due to the nature of our implementations requiring hands present in the image.

The dataset offers several advantages that align with the objectives of our static SLR pipeline and provides a solid basis for training and evaluating a CNN classifier. Due to the synthetic nature of the dataset, we can exercise precise control over various factors of the signs, including hand shapes, skin colour, image lighting, and backgrounds. The diversity of the dataset, coupled with the significant number of samples available for each letter, leads to the development of a robust model that can handle real-life conditions. Furthermore, the quality of the dataset is relatively high, with each of the images having a resolution of `512 x 512` pixels.

3.2.3 Custom Validation and Test Data

In addition to the Synthetic ASL Dataset, we also manually recorded a custom test dataset consisting of 20 images for each ASL letter. The purpose of this additional data was to validate the performance of the model in settings where the image originates from a standard laptop webcam and hands might be less clearly defined.

In order to capture the images, a signer who is fluent in ASL was told to perform each letter of the ASL alphabet under varying conditions, recording each image with their laptop webcam. Of the 20 images obtained for each class, we adopted the methodology of capturing 10 images in optimal conditions from a front-on angle with natural light and no occlusions. Meanwhile, 5 images from each class were captured from slightly varying camera angles, and a further 5 were captured in low light conditions, simulated using a lamp in a dark room. Each image then had the relevant portion of the hand cropped from the overall image (using the method described in Section 3.3.1) to obtain the final custom dataset.

3.2.4 Dataset Challenges and Limitations

Whilst the Synthetic ASL dataset provides many advantages for use in our static SLR pipeline, it is important to note that the synthetic nature of the dataset might result in simplified hand representations that do not fully represent the true nature of real-life hands. It is also unclear whether there is any bias as a result of the image generation process.

Our custom dataset also lacks the diversity of having multiple signers and is also relatively limited by its size. Whilst the dataset is only to be used for testing purposes, it is important to note the performance on the test dataset may not accurately reflect the performance of the model in the final platform.

3.3 An End-To-End Learning Approach to Static SLR

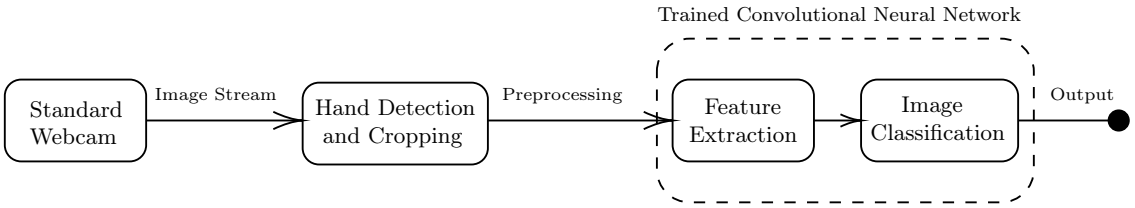


Figure 3.3: Our End-to-End Static SLR Pipeline structure

In this section, we present our initial attempt at designing an architecture for our static SLR pipeline. We employ an end-to-end learning approach [57], where we train a single deep neural network for image classification using the raw image data as an input (after some cropping and preprocessing) without any manual feature extraction. End-to-end learning has gained popularity due to the increased power of CNNs and the availability of abundant data. Networks that operate directly on pixel data are extensively used in image classification, eliminating the need for manual extraction of domain-specific features.

3.3.1 Proposed Hand Localisation Method

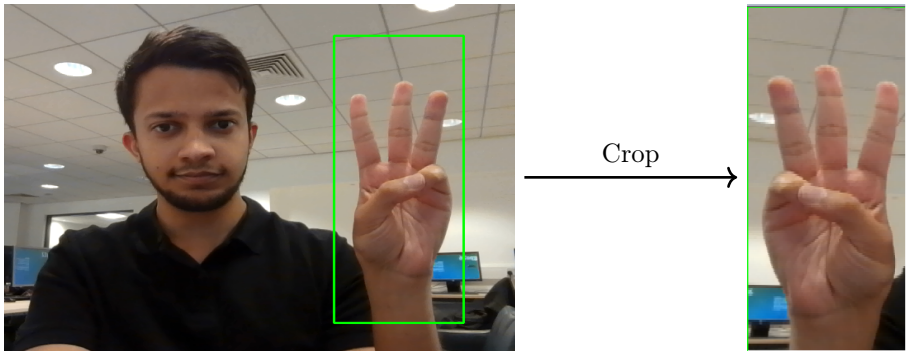


Figure 3.4: Cropping of hand the image using the bounding box defined by MediaPipe landmarks

As the initial step in our static SLR pipeline, we detect and localise the hand region from an input webcam stream. This allows for the cropping of an image of the user's hand, which can then be classified by the CNN.

We utilise OpenCV [50] in a Python script to open a user's webcam stream. Using MediaPipe's [51] Python library, we continuously extract hand landmarks from each individual frame of the input stream. The image is marked as non-writable during the detection process as a performance optimisation. To improve image quality and distinguish the hand from the background, we apply image preprocessing techniques, such as histogram equalisation and adaptive noise reduction filters from the OpenCV library. The extracted hand landmarks provide comprehensive information about the spatial dimensions of the hand.

Using the minimum and maximum x and y values from the landmarks, we can define a bounding box around the signer's hand, which serves as the region of interest. The coordinates defining the two opposite points of the bounding box rectangle, box_min and box_max, can be calculated as follows:

$$
\begin{aligned}
\langle x_{\min}, y_{\min} \rangle &= \langle \min(\text{landmark\_x}), \min(\text{landmark\_y}) \rangle \\
\langle x_{\max}, y_{\max} \rangle &= \langle \max(\text{landmark\_x}), \max(\text{landmark\_y}) \rangle \\
\langle \text{box\_width}, \text{box\_height} \rangle &= \langle x_{\max} - x_{\min}, y_{\max} - y_{\min} \rangle \\
\langle \text{padding\_x}, \text{padding\_y} \rangle &= \langle \alpha \cdot \text{box\_width}, \beta \cdot \text{box\_height} \rangle \\
\text{box\_min} &= \langle x_{\min} - \text{padding\_x}, y_{\min} - \text{padding\_y} \rangle \\
\text{box\_max} &= \langle x_{\max} + \text{padding\_x}, y_{\max} + \text{padding\_y} \rangle
\end{aligned}
$$

In the equations above, landmark_x and landmark_y represent the sets of x and y coordinates of the hand landmarks, respectively. The padding_x and padding_y values introduce a buffer around the hand landmarks, ensuring that the entire hand region is included within the bounding box. In our implementation, we set both α and β to 0.1.

3.3.2 Model Development and Architecture

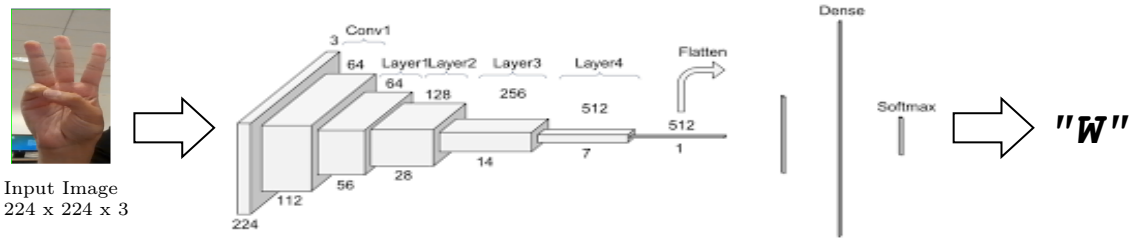


Figure 3.5: ResNet34 framework with a Softmax classifier [58] [59]

For our feature extraction and image classification CNN, we employ transfer learning [41] with the state-of-the-art ResNet34 Architecture [58]. The Resnet34 model has been pre-trained on over 100,000 images from the ImageNet database [42] and is commonly used for image classification. The model is composed of 34 layers, encompassing convolutional layers, residual blocks, and a global average pooling layer. The inclusion of residual blocks allows gradients to flow directly through skip connections, mitigating the vanishing gradient problem commonly encountered in deep neural networks. We use the Keras library [60] (version 2.10.0) to develop our model. The flow of data through our model is as follows:

1. We initialise the ResNet34 model with pre-trained weights and keep all but the last 3 layers frozen. The first few layers of a pre-trained ResNet34 capture low-level features such as edges, textures and other basic patterns that are applicable across many types of images. The unfrozen layers are trained and fine-tuned specifically for the SLR task. Freezing the majority of layers allows us to achieve enhanced training efficiency and also prevent overfitting issues.

2. We then down-sample the output of the ResNet's fully connected layers into a 26-node representation, with each node corresponding to a letter of the ASL alphabet.

3. Finally, we apply the Softmax activation function, commonly used in multi-class classification, to the 26-node output layer. After applying the Softmax function, the final output vector represents a normalised probability distribution over the classes, with each element corresponding to a probability of the input belonging to its respective class.

The overall architecture of our model (as shown in Figure 3.5) is summarised in Table 3.1:

Layer	Output Dimensions
Input Image	224 x 224 x 3
Stage 1 (Conv1)	112 x 112 x 64
Stage 1 (MaxPool)	56 x 56 x 64
Stage 2 (Residual Blocks)	28 x 28 x 128
Stage 3 (Residual Blocks)	14 x 14 x 256
Stage 4 (Residual Blocks)	7 x 7 x 512
Global Average Pooling	1 x 1 x 512
Fully Connected Layer	26
Softmax Classifier	26

Table 3.1: Summary of layers in our end-to-end architecture

3.3.3 Model Training and Tuning

We performed some minor preprocessing on the training data before training our ResNet34 model. To enhance training efficiency and maintain consistency with the ImageNet data used for pre-training, we resized all hand images to `224 x 224` pixels. Out of the 900 available training images for each ASL letter, we allocated 80% for training purposes and reserved the remaining 20% for validation. The validation set provides an unbiased measure of the model's accuracy and facilitates hyperparameter tuning.

We utilised a multi-GPU setup with two `NVIDIA Tesla P100` GPUs to accelerate training time via parallelisation. We trained the model using stochastic gradient descent and fine-tuned the model by exploring various combinations of hyperparameters. The set of hyperparameters that yielded the highest accuracy on the validation set was selected. To systematically explore different hyperparameters combinations, we used `GridSearchCV` from the Python library `scikit-learn` [61]. In Grid Search, we predefine a grid of hyperparameters, including learning rate, batch size, number of epochs, weight decay and the choice of optimiser and learning rate scheduler. Grid Search exhaustively trains the model on each combination of hyperparameters, returning a set of well-performing hyperparameter values. Due to constraints during training, we did not use cross-validation, which is a technique used to obtain a more robust estimate of the model's performance.

Hyperparameter	Best Value
Initial Learning Rate	0.001
Batch Size	32
Number of Epochs	50
Weight Decay	0.0001
Optimiser	Adam
Learning Rate Scheduler	ReduceLROnPlateau
Dropout Rate	0.2

Table 3.2: Hyperparameter Tuning Results

After our tuning, we obtained a set of hyperparameters, as shown in Table 3.2, to be used in the training of the model. We utilise the Adam optimiser, known for its efficiency and ability to handle sparse gradients, to update model parameters during training. The training is carried out for 50 epochs with an initial learning rate of 0.001. The ReduceLROnPlateau learning rate scheduler is used to monitor the validation loss to adjust the learning rate as necessary. In order to prevent overfitting to the data, We used a dropout rate of 0.2, randomly dropping out 20% of neurons

during training. We also employed early stopping if the validation accuracy remained similar for several epochs in order to avoid overfitting and save computational resources.

3.3.4 Model Evaluation

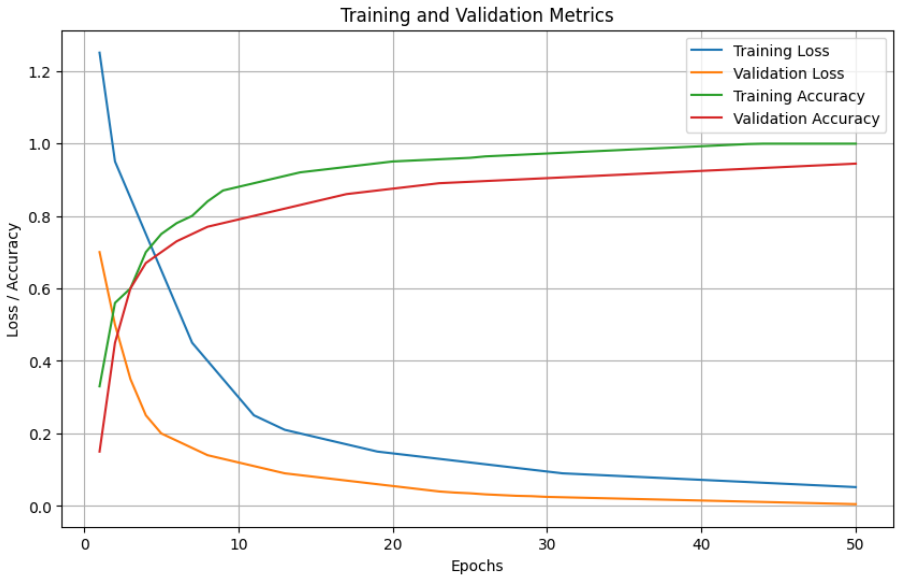


Figure 3.6: Loss and accuracy plots for training and validation sets over 50 epochs

To evaluate the performance of our end-to-end static SLR pipeline, we assessed the model via numerous metrics:

Training Analysis: Throughout our training process, we observed an improvement in both training and validation accuracies over 50 epochs, accompanied by a steady decrease in training and validation losses (see Figure 3.6). As expected, the validation accuracy begins lower than the training accuracy, gradually surpassing the training accuracy after a few epochs. The observed behaviour indicates that the model avoids overfitting to the training data, achieving an impressive final accuracy of 94.4% on the validation set after 50 epochs.

Classification Analysis: To assess the model's classification performance, we constructed a confusion matrix (shown in Figure 3.7) using the Synthetic ASL test set consisting of 100 samples from each of the 26 ASL letter classes. We use the Synthetic ASL Alphabet dataset to provide our test data. In the confusion matrix, each cell represents the percentage of predicted labels compared to the true labels. The diagonal elements indicate correct classifications, whilst off-diagonal elements represent a misclassification. The values have been normalised for better visualisation.

The confusion matrix provides us with valuable insights into the model's strengths and weaknesses and allows us to identify areas of potential improvement. The overall classification accuracy achieved on the test set, consisting of 2600 samples, was a promising 87.73%. However, the confusion matrix analysis reveals variations in the model's performance across different letters. Certain letters like "L" and "Y" achieve 100% classification accuracy, whilst others like "U" and "O" achieve much lower classification rates of around 68%. This is likely due to the distinctive hand shapes of letters like "L", making them easy for the model to interpret, whilst letters like "U" are much more similar to "R" and "V", with subtle variations in hand shape. Overall, our model's classification accuracy on the test set of 87.73% suggests a promising performance that aligns with most state-of-the-art static SLR pipelines. The classification accuracy could be further improved with additional hyperparameter tuning or training on a larger and more diverse dataset.

Real World Integration: However, we encountered more challenges when applying the model to our custom dataset, which included images recorded from a webcam in a very different setting to the Synthetic ASL dataset, but much closer to the environment that the SignSavvy app operates

in. The model's performance on this dataset significantly dropped, achieving only 61.2% overall classification accuracy. Additionally, when integrating the model into a full pipeline for real-time SLR, with images cropped by a MediaPipe bounding box, as discussed in Section 3.3.1, we observed a decrease in performance compared to offline testing. The model struggled to classify ASL letters in real-time on our extracted images, achieving an overall classification rate of less than 50%. The lower accuracy could be a result of various different factors. Although the training data encompassed a wide range of backgrounds and lighting conditions, the real-world images exhibited lower colour contrast compared to the dataset, posing a challenge for accurate classification. Furthermore, there were often cases where other parts of the signer's body, like their face, appeared in the background of the cropped image. Although we could obtain more clean custom test data with fewer occlusions, the noise accurately represented some of the issues the static pipeline will face when used in real-world settings as part of SignSavvy.

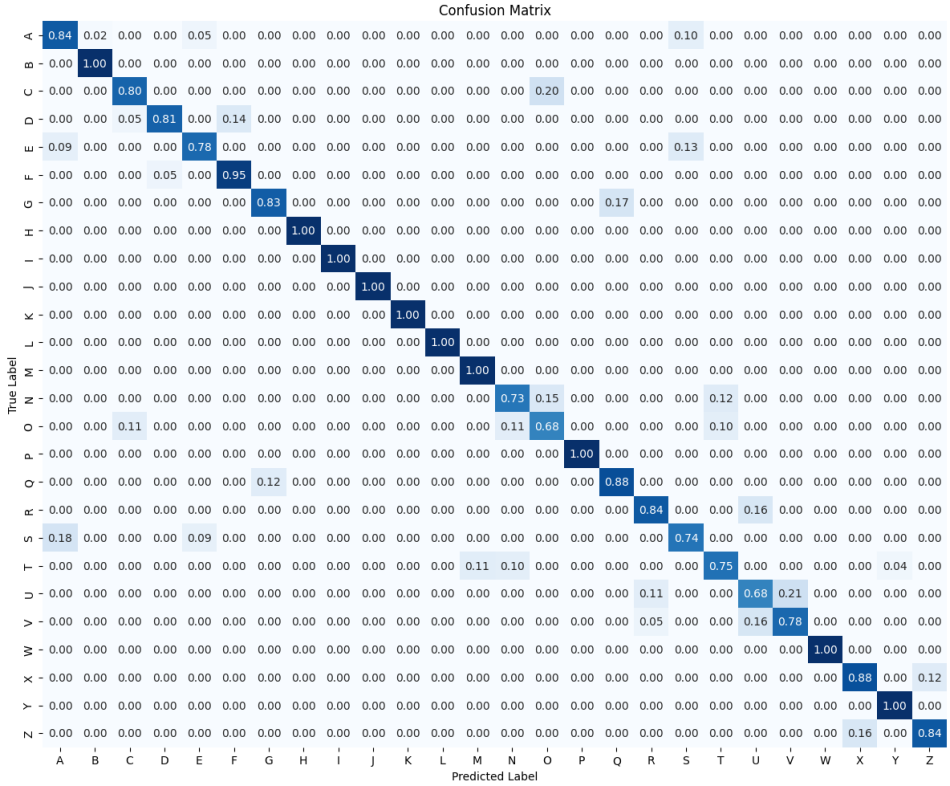


Figure 3.7: Confusion Matrix illustrating classification performance on Synthetic ASL test set

To address this accuracy discrepancy, we attempted to include our custom dataset in the training data. We also attempted some data augmentation of the original training images, applying random contrast changes and 10-degree rotation transformations. However, the classification performance on the connected SLR pipeline remained unsatisfactory for use in the application, with any captured image with less than ideal conditions resulting in an incorrect classification. In addition, the usage of a complex ResNet34 model resulted in an average classification time latency of 1.2 seconds. Whilst the latency is not prohibitive in our project, it is sub-optimal for a real-time application.

Conclusion: Our end-to-end static SLR model demonstrates promising overall results (see Table 3.3), achieving high accuracies on the test dataset. However, it also highlights challenges in correctly identifying certain letter pairs and faces some significant issues when adapted to use with real-world images.

Dataset	Training	Validation	Test (Synthetic)	Test (Custom)	Real-World
Accuracy (%)	99.9	94.4	87.7	61.2	< 50

Table 3.3: Final accuracy results for classification on different sets of data

3.4 A Feature-Based Approach to Static SLR

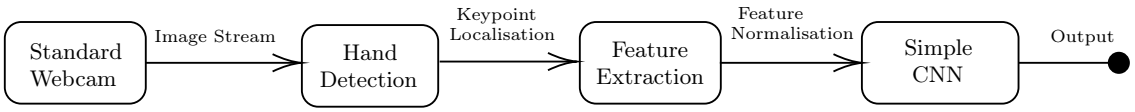


Figure 3.8: Our Feature-Based Static SLR Pipeline structure

In the previous section, we explored an end-to-end approach for static sign language recognition using a complex deep learning model. Despite the promising results achieved during testing, we encountered issues in accurately classifying ASL gestures, particularly in scenarios involving custom data and real-world applications when integrated with a full pipeline. To overcome the limitations encountered in our previous end-to-end approach for static SLR, we propose an alternative feature-based approach that focuses on manual feature extraction using a keypoint localisation tool like MediaPipe.

The feature-based approach aims to address the shortcomings observed in the end-to-end method by leveraging pre-trained models to extract meaningful hand features. Utilising MediaPipe's hand landmark estimation framework allows us to obtain accurate hand landmark predictions while also providing the added benefit of interpretability. In this section, we delve into the details of our feature-based approach to static SLR, discussing the process of extracting relevant features from the hand landmarks to be used as input to our custom classification CNN. We also highlight potential advantages and disadvantages of the approach compared to the end-to-end approach, particularly in the context of our custom test data and real-world integration. By adopting a feature-based approach and manually extracting the hand landmarks, we aim to enhance the accuracy and robustness of our static SLR system.

3.4.1 Proposed Hand Localisation Method

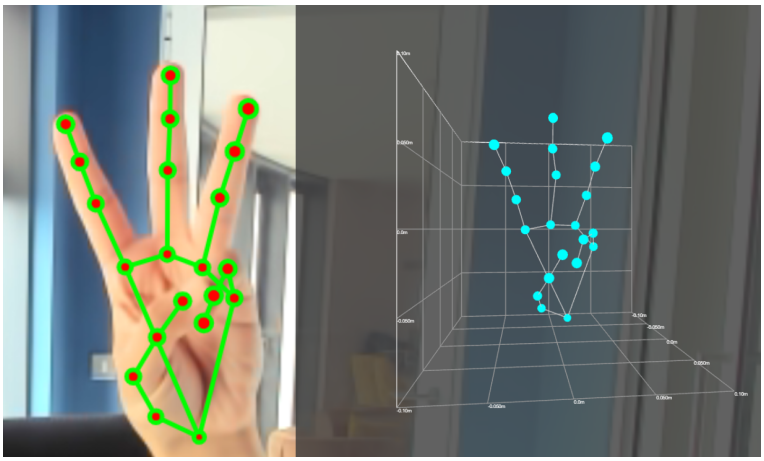


Figure 3.9: Extraction of hand landmarks for ASL sign "W" and 3d landmark representation

For hand landmark feature extraction, we use the MediaPipe Hands library, as detailed in Section 2.3.2. We utilise OpenCV in a Python script to open a user's webcam stream and MediaPipe's Python library, we continuously extract hand landmarks from each individual frame of the input stream. The image is marked as non-writable during the detection process as a performance optimisation. To improve the hand detection accuracy of the MediaPipe framework, we attempted to preprocess the image stream by applying noise reduction using the OpenCV library. Although adaptive histogram equalisation showed no noticeable improvement in hand detection and localisation, we observed a small improvement when using an adaptive filter to make the edges of the hand appear more defined.

The pre-trained MediaPipe hands model returns 3-dimensional coordinates of 21 key points of the hand, including fingertips, knuckles and the palm. It is important to note that the landmarks

provided by MediaPipe are normalised relative to the screen dimensions. To ensure consistent spatial representation of the landmarks, we normalise their coordinates relative to a bounding box encompassing the hand region, obtained through the method described in Section 3.3.1, with padding values α and β set to 0.

We utilise additional OpenCV and MediaPipe utilities for visualising and drawing landmarks. The overlay the extracted landmarks onto the input frames, annotating each landmark with a dot to highlight its location. The dots are connected to obtain a skeletal representation of the hand, which can then be extracted to create a 3d wire-frame visualisation (shown in Figure 3.9). The visualisations allow us to interpret the hand localisation process and assess the strengths and weaknesses of MediaPipe's framework.

Based on our experimentation, the MediaPipe Hands library demonstrates strong performance in various conditions, effectively handling challenges such as low-light environments and situations where the hand overlaps with other skin areas, such as the face. The library does face some issues in specific situations, such as when adjacent fingers are overlapping. In Figure 3.10, it can be observed that the index and middle finger overlap each other, but the MediaPipe library confuses the fingers and incorrectly denotes both as straight. The issue appears to occur when MediaPipe is configured to deal with a video input stream rather than an individual image. Despite the limitations, MediaPipe landmark extraction is generally very robust and accurate, allowing us to tackle some of the problems faced in real-world usage of our end-to-end pipeline.

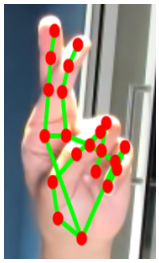


Figure 3.10: MediaPipe incorrectly representing adjacent overlapping fingers as straight

3.4.2 Model Development and Architecture

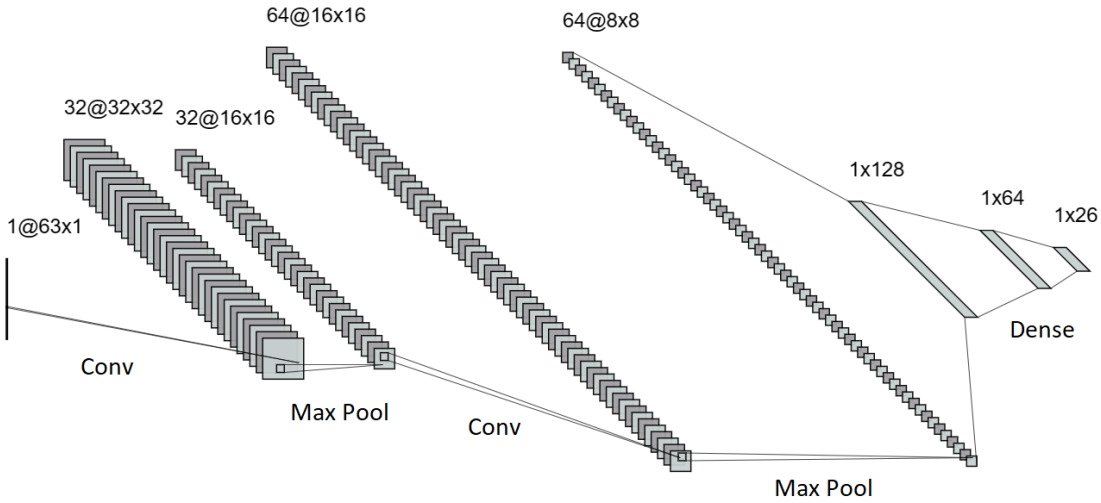


Figure 3.11: Our custom classification architecture operating on an 1 x 63 input feature vector

In our feature-based approach to static SLR, we leverage landmarks obtained from the MediaPipe framework, normalised relative to the bounding box defining the hand, rather than directly processing RGB pixel data like in our end-to-end approach. Overall, there are 21 3-dimensional coordinates outputted by the hand landmarker model, resulting in a flattened `1 x 63` tensor representation as

a feature vector.

For our classification task, we design a CNN model architecture that can effectively learn and classify the hand landmarks into a corresponding ASL letter class. We use the Keras library [60] to develop our model. Our custom classification model, as shown in Figure 3.11, consists of multiple convolutional, pooling and dense layers to extract meaningful features from the input hand landmarks. After each convolutional layer, we apply a batch normalisation layer to accelerate the training process and improve the overall performance of the model by reducing the internal covariate shift during training. We also apply a rectified linear unit (ReLU) activation function after each convolution and dense layer to introduce non-linearity and allow the model to learn complex representations.

After careful consideration, we determined the number of filters, kernel sizes, and layer sizes in our architecture based on empirical observations and previous research, aiming to strike a balance between model complexity and learning capacity. Specifically, the first layer of our model is a convolutional layer with 32 filters and a kernel size of 3x3. After applying batch normalisation and ReLU activation, this is followed by a max-pooling layer with a pool size of 2x2 to downsample the feature maps. We then include another convolutional layer, with 64 filters and kernel size of 3x3, and a further max-pooling layer. The resulting feature maps are then flattened into a 1D feature vector and passed into two dense layers of size 128 and 64, respectively. Finally, the second dense layer connects to an output layer consisting of 26 nodes, on which a Softmax activation function is applied. After applying the Softmax function, the final output vector represents a normalised probability distribution over the classes, with each element corresponding to a probability of the input belonging to its respective ASL alphabet class. A summary of our CNN model's layers can be found in Table 3.4.

No.	Layer	Output Size	Number of Parameters	Function
1	Input	63	-	-
2	Convolutional Layer 1	32x32x32	320	ReLU
3	Max Pooling Layer 1	16x16x32	-	-
4	Convolutional Layer 2	16x16x64	18,496	ReLU
5	Max Pooling Layer 2	8x8x64	-	-
6	Flatten	4096	-	-
7	Dense Layer 1	128	524,416	ReLU
8	Dense Layer 2	64	8,256	ReLU
9	Output Layer	26	1,690	Softmax

Table 3.4: Architecture and layers of our custom classification model

3.4.3 Model Training and Tuning

To allow for a fair comparison with our end-to-end model, we applied the same data preprocessing and dataset splits as detailed in Section 3.3.3, including resizing our training images to 224 x 224 and allocating 80% of the 900 available training images from the Synthetic ASL Alphabet dataset, with the remaining 20% to be used as a validation set. We also similarly use `GridSearchCV` to tune the model and obtain an optimal set of hyperparameters. Our GPU setup remains unchanged as we utilise two `NVIDIA Tesla P100` GPUs to accelerate training time via parallelisation.

After our tuning, we obtained a set of hyperparameters, as shown in Table 3.5, to be used in the training of the model. We use an Adam optimiser, known for its efficiency and ability to handle sparse gradients, to update model parameters during training. The training is carried out for 20 epochs with an initial learning rate of 0.001. We used a dropout rate of 0.4, randomly dropping out 40% of neurons during training. We also employed early stopping if the validation accuracy remained similar for several epochs in order to avoid overfitting and save computational resources.

Hyperparameter	Best Value
Initial Learning Rate	0.001
Batch Size	64
Number of Epochs	20
Weight Decay	0.0001
Optimizer	Adam
Learning Rate Scheduler	None
Dropout Rate	0.4

Table 3.5: Hyperparameter Tuning Results

3.4.4 Model Evaluation

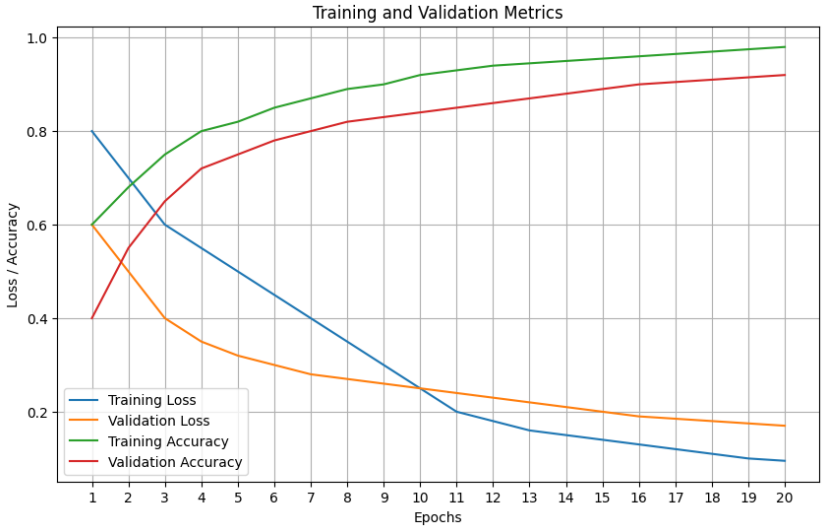
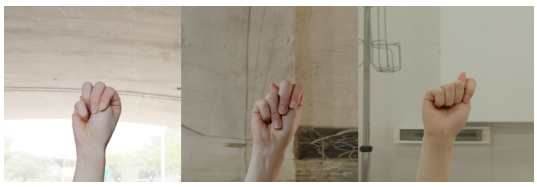


Figure 3.12: Loss and accuracy plots for training and validation sets over 20 epochs

Similar to the evaluation of our end-to-end model, we evaluate the performance of our model based on various metrics:

Training Analysis: The training and validation curves (Figure 3.12) show a similar pattern to those of our end-to-end model, achieving gradually improved training and validation accuracies, but over 20 epochs rather than 50. The steady decrease in training loss suggests that the model is effectively learning from the training data, while the consistent decrease in validation loss indicates its ability to generalise well to unseen data. The feature-based model achieves a final accuracy of 91.6% on the validation set after 20 epochs, which is in line with the performance of the end-to-end model. The plateauing effect of the curves indicates the model is near its optimal performance and that training for more epochs might cause overfitting.

Classification Results: Our final model achieves an ASL alphabet classification accuracy of 90.3% on the 2,600 images from the Synthetic ASL Alphabet test dataset. This signifies the model's capability to effectively classify ASL letters on data prepared in a similar way to the training data. Furthermore, the performance on our custom dataset, made of images captured by a webcam, achieves an impressive classification accuracy of 89.8%. This performance is considerably better than that of our end-to-end approach, which achieved an accuracy of only 61.2% on the custom dataset. The higher accuracy on our custom dataset indicates our feature-based model is much more robust to variations in images and deals much better with the nuances of non-generated photos that feature noise and other occlusions like faces and varying angles.



(a) ASL Letters "M", "N" and "T" respectively

(b) ASL Letters "L", "Y" and "W" respectively

Figure 3.13: Samples from Synthetic ASL Dataset that are classified poorly (left) and well (right)

Classification Analysis: To further assess the strengths and weaknesses of the model's classification capability, we generate another confusion matrix (shown in Figure 3.14). Instead of using the Synthetic ASL Alphabet test data, we instead plot the matrix from the results on our custom dataset of 520 images to better assess the model's performance on real-world images. The diagonals of the confusion matrix represent correct classifications, whilst off-diagonal elements represent misclassifications. The values have been normalised for better visualisation. From the confusion matrix, it can be observed that the model achieves almost perfect results for many letters, including "L", "Y", and "W", which all feature very distinct hand shapes with very little finger overlapping (see Figure 3.13b). Meanwhile, certain groups of signs suffer from being mixed up by the model. For example, "R" is often mixed up with "U" and "V", despite having a distinctive finger overlap of the index and middle finger. One explanation for this is a shortcoming of MediaPipe discovered in our experimentation, where the model struggles to accurately pick up finger overlaps (this can be seen in Figure 3.10, where the letter "R" is being signed but the visualised skeleton appears closer to the letter "U"). The model also often confuses the letters "M", "N", and "T", which all feature a closed fist with the thumb between different fingers (see Figure 3.13a). These similarities make it challenging for a model to accurately differentiate between them, with the end-to-end model operating on visual features also struggling with this group. These letters also feature significant finger overlap, which may be a factor contributing to the loss in accuracy.

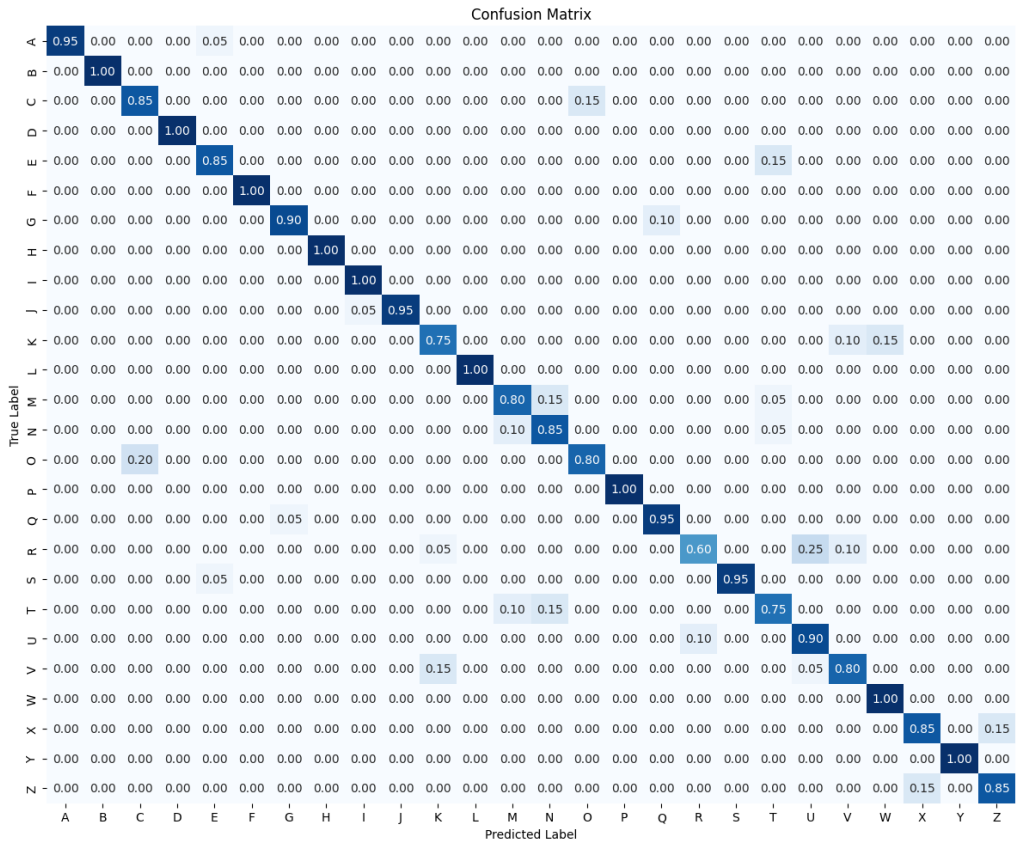


Figure 3.14: Confusion Matrix illustrating classification performance on our custom dataset

Performance Analysis: To assess the performance of our model in our full pipeline, we create a Python application that extracts hand landmarks from a continuous web stream. At each frame, the image is provided to the MediaPipe library, which outputs the coordinates of landmarks as detailed in Section 3.4.1. We then perform the necessary conversions of the features using the NumPy library [62] and use the features as input to the saved custom Keras model. The classification result of the model is then printed to the console. To evaluate overall accuracy, a signer fluent in ASL attempted to perform the 26 letters of the ASL alphabet. The signer was instructed to perform the letters as naturally as possible to simulate a realistic use case of ASL. Of the 26 signs, only the letters "K", "R", and "T" returned the wrong classification, instead being identified as "V", "U", and "N", respectively, resulting in an overall classification accuracy of 88.5% in our real-world pipeline. It was found that the signer had to exaggerate these signs into unnatural positions in order for the system to correctly identify them.

One notable advantage of our feature-based model is its parameter efficiency, with only 553,178 trainable parameters. Meanwhile, the deep ResNet34 model employed by our end-to-end approach consists of over 20 million parameters. The simplicity of the model architecture in our feature-based approach compared to the end-to-end approach is attributed to the nature of the input data. Using the hand landmarks, we are able to directly capture the key spatial information of the hand shape, eliminating the need for complex convolutional layers to extract high-level image features. The reduced model complexity enables faster training and inference times, with an average classification time latency of only 224 milliseconds. In comparison, our end-to-end model took 1.2 seconds using the same GPU setup. This makes our feature-based approach more suited to real-time applications such as SignSavvy that require classification on a continuous stream of images. The parameter efficiency also results in a more compact and portable model that is easier to deploy, as well as a reduced risk of overfitting to training data. Further work is required to evaluate if even simpler architectures could achieve improved performance, given the limited input feature vector size.

Conclusion: Overall, our feature-based approach provides a promising solution for a static SLR pipeline to be used in SignSavvy. The model shows a significant improvement in accuracy and efficiency compared to our end-to-end approach, especially on real-world images that we collect from webcams. Despite the significant advancement, one limitation of the approach is that it relies on the accurate hand localisation and landmark extraction of the MediaPipe Hands library. Assuming the MediaPipe library continues to develop, the performance of the model should accordingly improve in the future. Our final results on unseen test data are comparable to that of the state-of-the-art in real-time static SLR, showing an impressive final result considering limitations in the library and compute resources available for training and tuning.

Dataset	**Training**	**Validation**	**Test (Synthetic)**	**Test (Custom)**	**Real-World**
Accuracy (%)	98.2	91.6	90.3	89.8	88.5

Table 3.6: Final accuracy results for classification on different sets of data

Chapter 4

Dynamic SLR Model Implementation

In this chapter, we present the implementation details of the dynamic sign language recognition pipeline, designed to analyse and classify dynamic ASL signs captured in video format. Building upon the principles and techniques discussed in Section 2.2, we have developed a customised approach specifically tailored to address the unique challenges of dynamic sign recognition. This chapter provides an in-depth exploration of the various components involved in our dynamic SLR pipeline, including data collection, landmark extraction, feature representation and similarity matching. Throughout this chapter, we will discuss the intricacies of our implementation, highlighting the key methodologies we use to achieve robust and efficient dynamic sign recognition.

4.1 Objectives and Proposed Methods

In this section, we clearly define the motivations behind the implementation of our dynamic SLR pipeline, as well as the specific requirements that our designed system should tailor to. Through this, we establish a framework for our subsequent exploration into potential solutions:

1. Development of a dynamic SLR pipeline that can accurately classify a diverse set of dynamic ASL gestures, including signs with subtle hand/pose variations and temporal dynamics.
2. Development of a robust system that can effectively operate on low-resolution videos and handle real-world challenges, such as variations in lighting conditions, skin colour, backgrounds, camera angles, and occlusions.
3. Development of a dynamic SLR system that can adapt to perform near-real-time classification of signs from continuous live streams. We will explore various performance optimisations to ensure efficient processing and classification.
4. Design of a robust feature vector representation that is tailored specifically to the spatiotemporal nature of dynamic signs and is invariant to factors such as camera zoom and sign speed.
5. Evaluation of the robustness, efficiency, and scalability of the dynamic SLR pipeline, considering metrics such as classification accuracy, inference time, and resource utilisation. This evaluation will ensure the practical viability of the pipeline in real-world scenarios.

4.2 Data Selection and Composition

In this section, we discuss the process of data selection and composition for our dynamic SLR pipeline. We also discuss how we extract landmarks from videos in an efficient manner and store them in a dataset for later use.

4.2.1 WLASL Dataset and Video Data Collection

The Word-Level American Sign Language (or WLASL) Dataset [27] is the largest video dataset of whole ASL words, consisting of around 2,000 different commonly used ASL words. In total, the dataset consists of 21,083 videos performed by 119 signers. We choose this dataset over other dynamic ASL datasets like Purdue RVL-SLLL [23] due to the significant amount of instances per class available and the large range of signers, providing a solid foundation for the basis of our dynamic model. The dataset's videos are primarily sourced from two sources: educational websites like ASLU [63] and ASL-LEX [21], and also YouTube tutorial videos. In each video, a signer performs only one sign in a nearly-frontal view with different backgrounds.

Specifically, we use version v0.3 of the WLASL dataset, released in March 2020. For each video, the WLASL provides a corresponding gloss, which serves as an English translation of the signs performed in the video. This provides a consistent and standardised way to identify and label signs in the dataset. In addition to providing a gloss label for each video, some metadata information, including temporal boundary, body bounding box, signer annotation and sign dialect/variation annotations, are provided by the dataset. This additional metadata is ignored for our project as it is not directly relevant to our proposed implementation.

The WLASL dataset also includes a JSON file that contains the gloss annotations, video URLs and metadata for each video. We parse the file using a Python script and adapt a bundled downloader script to download all available videos for a specified set of signs. Specifically, we choose the following set of words as a starting point for our exploration into dynamic SLR:

- Yes
- No
- Please
- Thank you
- Sorry
- Goodbye
- Hello
- How are you?
- Help
- Go
- Drink
- Chair

The selection aims to cover a range of sign complexities and hand configurations, providing a good basis for an exploration of our dynamic SLR pipeline. The chosen set of words encompasses a variety of signs performed with one hand, like "yes" and "please", as well as signs that require two hands to perform, like "go" and "chair". Furthermore, we select a subset of the most frequently used words in daily interactions, with the majority of our words included in lists from numerous ASL websites of the most important phrases to learn as a beginner [64].

It should be noted that signs like "please" and "sorry" are performed very similarly, with both involving a single hand moving in a circular motion near the chest. However, "Please" is performed with fingers extended, whilst "Sorry" is performed with the hand forming a fist. We intentionally include such similar signs to evaluate the capability of our dynamic SLR pipeline in capturing subtle nuances and distinguishing between visually similar signs.

For each of our selected signs, we choose a collection of 10 videos, ensuring each video comes from a different source and signer. This approach allows us to gather diverse examples of each sign, but we take special care to ignore videos that demonstrate substantial variations of a sign to avoid any potential confusion in the system.

4.2.2 Landmark Extraction and Storage

In order to extract landmarks from videos, we use the MediaPipe library [6], as detailed in Section 2.3.2. MediaPipe provides dedicated solutions for pose and hands tracking and landmark localisation, namely MediaPipe Pose and MediaPipe Hands. The process is summarised as follows:

1. To streamline the landmark extraction process, we define a custom method to save landmarks from a video. This function facilitates the extraction of all landmarks from a specified video. It operates by first loading the specified video and converting it into a source stream using OpenCV [65]. Each frame of the video is handled individually, with MediaPipe Hands landmark

extraction performed for the hand, or both hands if present, and MediaPipe Pose landmark extraction for body landmarks like the shoulders, nose and eyes. Instead of extracting all the landmarks from the pose model, we choose to only select landmarks 0-22, ignoring all pose landmarks from the hips and below. We choose to ignore landmarks like knees, feet and hips as they would typically not be visible in a standard laptop webcam or phone camera setup.

2. The MediaPipe framework returns 3-dimensional coordinates of the relevant landmarks relative to the overall video dimensions. During the extraction process, the landmarks for the left hand, right hand and pose are obtained as separate lists. For a given time frame, there are a total of 63 data points (21 landmarks in 3 dimensions) recorded for each hand and 39 data points (13 landmarks in 3 dimensions) recorded for the pose. In the case where a given hand has no detection for a specific time frame, an empty list is instead used to represent the hand.

3. The three lists of landmarks for the left hand, right hand, and pose are extracted and aggregated from each frame of the video, resulting in a final list of lists with dimensions $N \times M$, where N represents the number of frames in the video and M represents the total number of data points per frame (63 for hand landmarks and 39 for pose landmarks). The resulting list of lists serves as a compact representation of the extracted landmarks, allowing for efficient storage and further analysis in later stages of our dynamic SLR pipeline. We can represent the structure of the extracted landmark for a hand or pose data of a video as follows:

$$\begin{bmatrix} \text{Frame 1} & [L_{11}, L_{12}, \ldots, L_{1M}] \\ \text{Frame 2} & [L_{21}, L_{22}, \ldots, L_{2M}] \\ \vdots & \vdots \\ \text{Frame N} & [L_{N1}, L_{N2}, \ldots, L_{NM}] \end{bmatrix}$$

   where L_{ij} denotes the j-th landmark data point in the i-th frame.

4. After aggregating the landmarks, we store the resulting list of lists in the pickle format. Pickle is a Python library that allows for easy serialisation and deserialisation of Python objects, which also enables us to store binary files of our representations for later use.

We define a general function for updating the dataset, taking two arguments: A video dataset folder containing all the videos of signs, organised into subfolders corresponding to the gloss of the sign, and an output pickle dataset folder, which stores 3 separate pickles (corresponding to left hand, right hand and pose) for each video, organised in the same subfolder structure as the input folder. To avoid unnecessary computation, we optimise this function by providing an overwrite flag. When the overwrite flag is set to false, videos that already have pickles present in the output folder and skipped.

4.3 Dynamic SLR Pipeline Architecture

In this section, we delve into the architecture and implementation of our dynamic SLR pipeline, including the key components that make up the system, as well as how we ensure robust representation of features. We also discuss our process for loading the stored dataset, created as discussed in Section 4.2.2.

4.3.1 HandModel and PoseModel

The HandModel and PoseModel are two fundamental classes within our Python implementation of the dynamic SLR pipeline, responsible for capturing and representing the hand and pose information of an image, respectively. Each instance of the HandModel represents an individual hand at a specific time frame. Similarly, each PoseModel represents a signer's pose at a specific time frame. Upon initialisation of a HandModel or PoseModel object, taking an input of a hand list (of length 63) or pose list (of length 39), the model initialises a feature vector to be used for similarity comparison and classification in later stages.

One of the challenges encountered when using raw MediaPipe extracted landmarks as classification features is their sensitivity to factors such as size variations and absolute positions of hands and the body. It is important to note that MediaPipe returns landmark coordinates relative to the screen or video dimensions, resulting in very little utility without some manual normalisation or adjustment. Therefore, we introduce a feature vector that mitigates the influence of size and position while still preserving the inherent structure of the hand and pose. We calculate the angle between each of the connections in the hand or pose, with connections being defined as shown as the green connecting lines in Figure 2.9a and Figure 2.9b (above and not including the waist only).

The MediaPipe library offers a convenient feature of providing a predefined set of connected landmark indices for both hand and pose connections. These connections are represented as 2-element tuples, allowing us to establish geometric relationships between landmarks. Using the predefined set, we can define a geometric vector for each connection, formed by the corresponding 3-dimensional landmark coordinates. We then calculate the angles between the computed geometric vectors. An example of a pair of connections and the angle between them is shown in Figure 4.1. The angle calculation is performed in NumPy [62] using the dot product formula as follows:

$$\theta = \arccos\left(\frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\|}\right)$$

where θ represents the angle between two vectors $\mathbf{v_1}$ and $\mathbf{v_2}$. By calculating these angles for every connection, we obtain a final feature vector that captures the relative geometric information of the hand or pose. This feature vector serves as a robust representation of our dynamic SLR model.

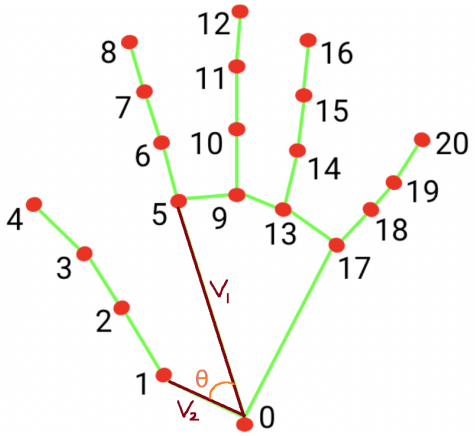


Figure 4.1: Diagram of feature angle and geometric vectors for hand connections (0,5) and (0,1)

The HandModel also takes an input of the handedness of the corresponding hand (left or right hand) as an input on initialisation. The MediaPipe library also conveniently returns handedness as part of its hand landmark extraction process.

4.3.2 DynamicModel Composition and Creation

The DynamicModel class is a key component of our dynamic SLR pipeline, responsible for integrating the pose model and one to two hand models for each time frame in a video sequence. By integrating the components together, we can generate a representation of changes in hand and pose over an entire video.

The full pipeline, from data loading to DynamicModel creation, is shown in Figure 4.2 and can be summarised as follows:

1. **Dataset loading**: We begin by loading the extracted landmarks of each video of our dataset, which were stored in the form of pickle files as described in Section 4.2.2. To facilitate further processing, we convert the lists into flattened NumPy arrays.

2. **DynamicModel initialisation**: We create a DynamicModel object for each video, passing in the video ID during initialisation. The DynamicModel class features three internal embedding lists, one for each hand and one for pose (as represented in Figure 4.2). Each index in the embedding lists corresponds to a specific time frame within the original video.

3. **Frame-by-frame feature extraction:** For each frame of the loaded video, we use the extracted landmarks to create the HandModel and PoseModel objects described in Section 4.3.1, along with our angle feature vector representation. For time frames where no hand data is available, a value of `None` is stored as a placeholder to ensure consistent indexing across all time frames. We repeat this process for every video to generate multiple DynamicModel objects.

4. **User recording of sign to be classified:** We create a Python application that uses OpenCV to display the webcam stream to the user. When the user clicks the "r" key on their keyboard, a 5-second video is recorded. At each time frame of the video, we extract the hand and pose landmarks, creating a DynamicModel object with HandModel and PoseModel object embeddings in a similar procedure to our dataset videos.

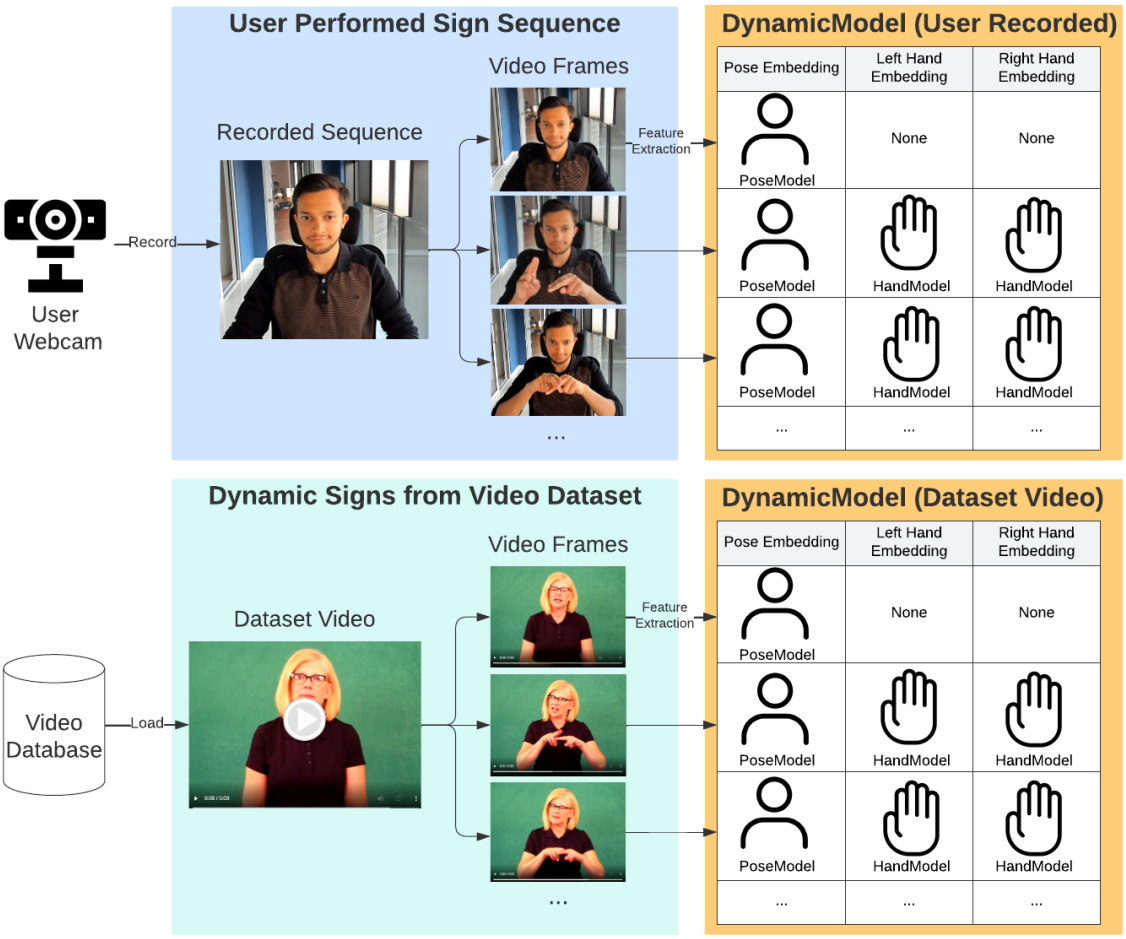


Figure 4.2: Creation of DynamicModel objects for frames of user recorded and dataset videos

4.3.3 Similarity Comparison and Classification Procedure

To classify the sign, we choose to use a similarity-based approach, comparing the recorded sign to each reference sign in the dataset to find the "closest" matches. Our DynamicModel objects represent the temporal and spatial information from each video and the recorded sign. In order to accurately compare the recorded sign features to the reference signs, we apply the dynamic time warping (DTW) algorithm (detailed in Section 2.2.8), using the `fastdtw` library in Python. Using

the DTW algorithm, we are able to compare signs with a different number of video frames, as well as signs that are performed out-of-sync at different speeds.

For each of our reference signs, we obtain a DTW distance, quantifying the dissimilarity between the sign to be classified and the sign from the dataset. Specifically, for two DynamicModel objects being compared, we apply DTW to each of the three embedding lists separately before summing up each distance to obtain an aggregated DTW distance (as shown in Figure 4.3). `None` values are omitted from their embedding list in the comparison.

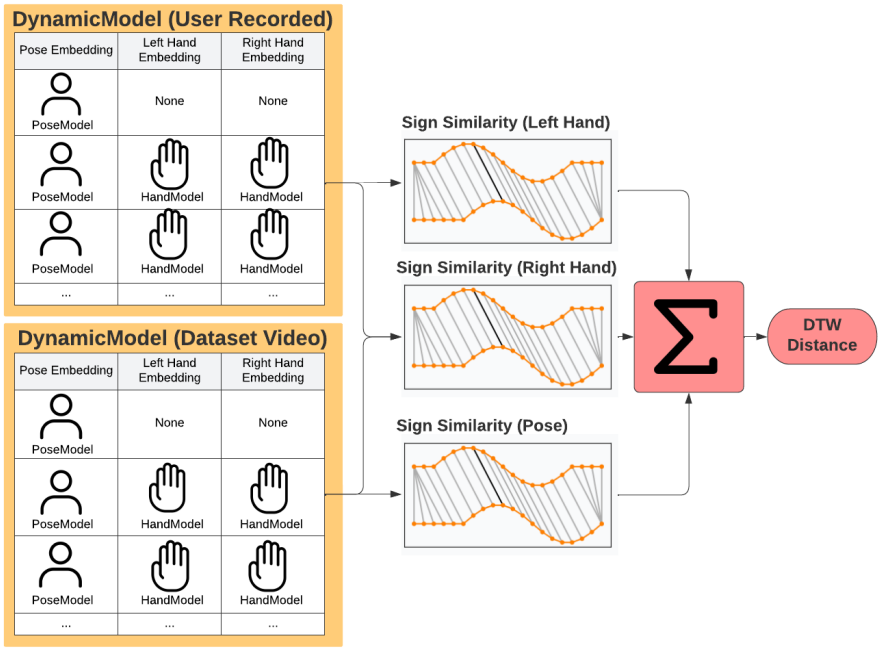


Figure 4.3: Similarity comparison for user recorded video and dataset video

We compute a dataframe table of distances by comparing the recorded sign with each video in the dataset. As an optimisation, we skip calculating DTW distances for signs that have a different number of hands present compared to the recorded sign. In such cases, we set the DTW distance to ∞. The dataframe is then sorted in ascending order based on the calculated DTW distances.

For the classification of the recorded sign, we perform batch classification by examining the first 10 reference videos from the sorted dataframe, which correspond to the smallest DTW distances). Within this batch of videos, we look for the most frequently occurring gloss or sign. To ensure robustness in the classification process, we require the most common gloss to appear at least four times within the batch for successful classification. This batch classification approach allows us to leverage the strength of collective evidence from the closest reference videos, increasing the reliability and accuracy of the classification process.

4.4 Evaluation

The evaluation of the dynamic SLR model's performance is crucial to assess its classification accuracy, robustness and scalability. In this section, we present a comprehensive evaluation of the pipeline, considering various metrics and methodologies:

Accuracy evaluation methodology: To assess the classification accuracy of our model, we conduct an evaluation using a test dataset consisting of 10 additional videos for each of the 12 sign classes. The additional videos are sourced using the same procedure as the reference dataset (detailed in Section 4.2). We calculate the classification accuracy by comparing the predicted sign labels generated by our model with the ground truth labels of the test videos. The accuracy was computed as the ratio of correctly classified signs to the total number of signs in the test set.

Classification Analysis: Of the 120 videos in the test set, 104 were classified correctly, giving a final classification accuracy of 86.7%. Analysis of misclassifications suggested signs with little hand variation that rely mainly on pose changes, such as "Thank You", were misclassified much more frequently than signs like "Chair", with much hand variation. Our initial implementation of the model only consisted of the PoseModel, without any fine-grained extraction of hand landmarks, correctly classifying only 61.7% of the signs on the test set.

To evaluate the robustness of the model, we examined cases where the top-ranked video from the similarity comparison did not correspond to the correct classification. Interestingly, our analysis revealed that out of the 97 correctly classified signs, only 64 had the correct sign gloss as their top-ranked video. This observation emphasises the significance of our batch classification approach, which considers multiple closest videos for accurate classification. It also indicates that the model occasionally produces anomalous similarity-matching results, suggesting a more advanced method of comparing similarity could improve results.

Scalability Analysis: In addition to the model's classification performance, we evaluated the scalability of our model. In order to evaluate how the model performs with more signs, we progressively increase the number of sign classes in our reference dataset. For each new sign, we obtain an additional 10 videos for reference comparison and 10 videos for testing, downloading using the script described in Section 4.2. We measure the impact on accuracy rates and the computational time required for classification. We measure the performance on the same machine (Windows 10, 11th Gen Intel Core i7-1165G7 Processor @ 2.80GHz, 32GB RAM) to ensure a fair comparison. This scalability assessment allows us to gauge the model's performance and generalisation ability as the dataset size expands, providing valuable information for practical deployment scenarios with larger datasets and its potential for handling a broader range of signs beyond the original set.

The results, as shown in Table 4.1, indicate that as the dataset size expanded, the accuracy rates remained consistently high, showing that our model can effectively handle larger datasets without sacrificing performance. However, it is important to note that the computational time for classification increased linearly with the dataset size, suggesting potential performance challenges when scaling the model to extremely large datasets with many supported signs. Further optimisations may be necessary to improve efficiency in such scenarios. As the dataset grew in size, we also faced numerous challenges in managing and efficiently processing a larger database of sign videos. These challenges led us to consider strategies for database management to effectively store pickle data.

Number of Sign Classes	Classification Accuracy (%)	Average Classification Time (s)
12	86.7	1.32
18	82.2	1.78
24	78.8	2.35
30	75.3	3.35

Table 4.1: Evaluation of our dynamic SLR pipeline's scalability for a broader range of signs

Conclusion: Overall, the evaluation of our dynamic model yielded promising results, with an 86.7% classification accuracy for a set of 12 different dynamic signs and an average classification time of just 1.32 seconds on our setup. These results demonstrate the effectiveness of our non-complex model, which achieves comparable accuracy to more advanced dynamic SLR methods, such as those based on deep Hidden Markov Models (HMMs), despite using a limited number of training samples. By avoiding the need for deep learning models, our approach allows for a lightweight solution to dynamic SLR, suitable for real-time applications, that can be easily expanded to include other signs without the need for huge amounts of training data. However, it is important to acknowledge the limitations of our current approach, particularly in accurately classifying signs with little hand variation, and also potential performance issues when scaled to hundreds of signs. To address these challenges and improve the model's robustness and scalability, further optimisations should be explored, such as integrating detailed facial landmarks or exploring the combination of deep learning models, such as HMMs, with dynamic time warping for improved temporal modelling.

Chapter 5

Building SignSavvy

In this chapter, we delve into the development and design process of SignSavvy, an educational platform for ASL that provides a comprehensive learning experience for beginners in sign language. Our platform aims to not only validate learners' sign performance but also offer personalised feedback and targeted suggestions for improvement, simulating the guidance of a real-life tutor. Moreover, to ensure widespread accessibility, our system will be designed to work seamlessly with commonly available hardware, such as phone cameras or laptop webcams, eliminating the need for specialised equipment or sensors. SignSavvy combines the advanced computer vision SLR pipelines discussed in previous chapters with intuitive user interfaces and targeted multi-modal feedback to help learners improve.

Throughout this chapter, we will discuss the key aspects of SignSavvy's design and development, highlighting the challenges faced and the solutions implemented. We will also showcase the architecture and deployment of the platform, exploring the technologies and frameworks that form its foundation. Finally, we will explore mechanisms and features that make SignSavvy an effective teaching tool, including our methods for demonstrating signs and multiple forms of image and text-based feedback.

5.1 Design, Development and UI

5.1.1 Architecture and Deployment

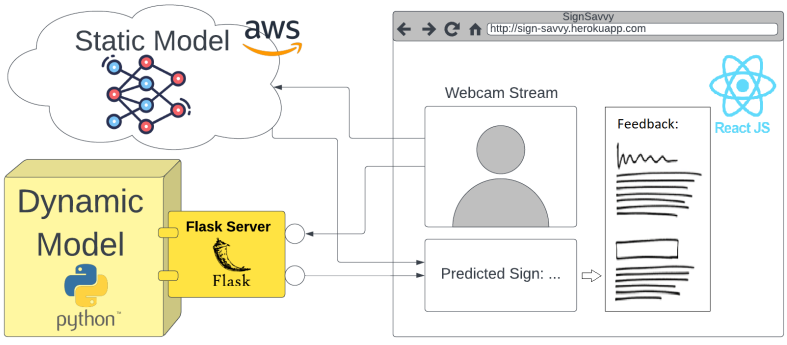


Figure 5.1: Architecture and Technology Stack for SignSavvy Platform

In the design and development of SignSavvy, the choice of a browser-based application was driven by our need for accessibility and ease of use, as well as ease of access for evaluation purposes. By adopting a web-based approach, SignSavvy eliminates the barriers of platform dependencies and allows users to access the application from any device with a modern web browser. The SignSavvy architecture consists of a combination of backend, frontend and cloud components that work together to create the platform (shown in Figure 5.1):

User Interface: SignSavvy features a user-friendly and intuitive React-based user interface (UI) on the frontend. The UI enables smooth user interaction and provides relevant sign information and feedback on the user's performance (further details in Section 5.1.2).

Deployment and Load Balancing: SignSavvy is hosted and deployed on Heroku, a cloud-based platform that offers scalability and ensures public accessibility through a dedicated web URL. We use Heroku's built-in load balancing capabilities to automatically distribute incoming requests across multiple dynamically created server instances, ensuring efficient resource utilisation and responsive performance. Heroku also provides monitoring capabilities for tracking system performance and status. Additionally, HTTPS encryption is implemented to secure the communication between the client and server, safeguarding user data and maintaining privacy.

Recording Component and MediaPipe Integration: The recording component in SignSavvy's frontend UI integrates with the user's webcam to facilitate sign recording. We use MediaPipe's [6] JavaScript library to perform direct extraction of pose and hand landmarks from the webcam stream. This approach replaces the method mentioned previously in our SLR pipelines, of using OpenCV [50] and Python for recording the user's sign and extracting landmarks.

Dynamic Model Integration and Flask Server: SignSavvy incorporates a Flask server on the backend, serving as an intermediary between the frontend and the Python-based dynamic SLR model (discussed in Chapter 4). By offloading the computationally intensive dynamic model to the server, we achieve efficient processing and seamless integration with the existing dynamic SLR pipeline. The server architecture includes endpoints for receiving aggregated extracted landmarks from the frontend and forwarding them to the dynamic model for classification. Another endpoint handles the posting of the classification result back to the frontend. To optimise communication overheads, we perform data aggregation over multiple time frames on the frontend before sending the landmarks to the backend.

Static Model Integration and AWS Deployment: We deploy our trained feature-based CNN (from Section 3.4) to AWS (Amazon Web Services). Deploying the model to the cloud ensures the availability and scalability of requests. During web app initialisation, the trained model is downloaded, allowing real-time classification by passing the extracted hand landmarks through the model to obtain an ASL letter result. SignSavvy considers a specific letter to be recorded when the same letter is classified for 10 consecutive frames. Caching mechanisms are implemented to avoid repetitive model downloads, resulting in faster response times and improved system efficiency.

Feedback Component: SignSavvy's feedback component plays a vital role in providing an interactive and engaging learning experience. The feedback loop begins with the demonstration of a sign the user should perform (more details in Section 5.1.3). After the user performs the sign and the respective model returns its classification result, we generate feedback by comparing the expected sign with the sign classification. If the classifications match, positive feedback is displayed to the user; otherwise, a more detailed analysis of the geometric differences between the recorded and expected sign is performed on the client side. This is then presented to the user in many forms of detailed feedback, including hints to improve, sign playback and visual annotations (more details in Section 5.2).

5.1.2 User Interface Development and Design

The user interface (UI) development and design of SignSavvy play a crucial role in providing a seamless and accessible experience for users. The UI is designed to be intuitive, visually appealing, and user-friendly, ensuring effective communication and interaction with the application. Some of the key aspects of our UI design can be summarised as follows:

Interactive and Responsive Design: The UI of SignSavvy is designed to be responsive and cater to a wide range of devices and screen sizes, further increasing the application's accessibility. The responsive design ensures that the application adapts seamlessly to different platforms, including desktops, laptops, tablets and mobile devices. The UI incorporates interactive elements, such as buttons and controls, that respond to user inputs in real time. For example, when users record their sign, the recording window visually responds to the input, and the recording button turns red

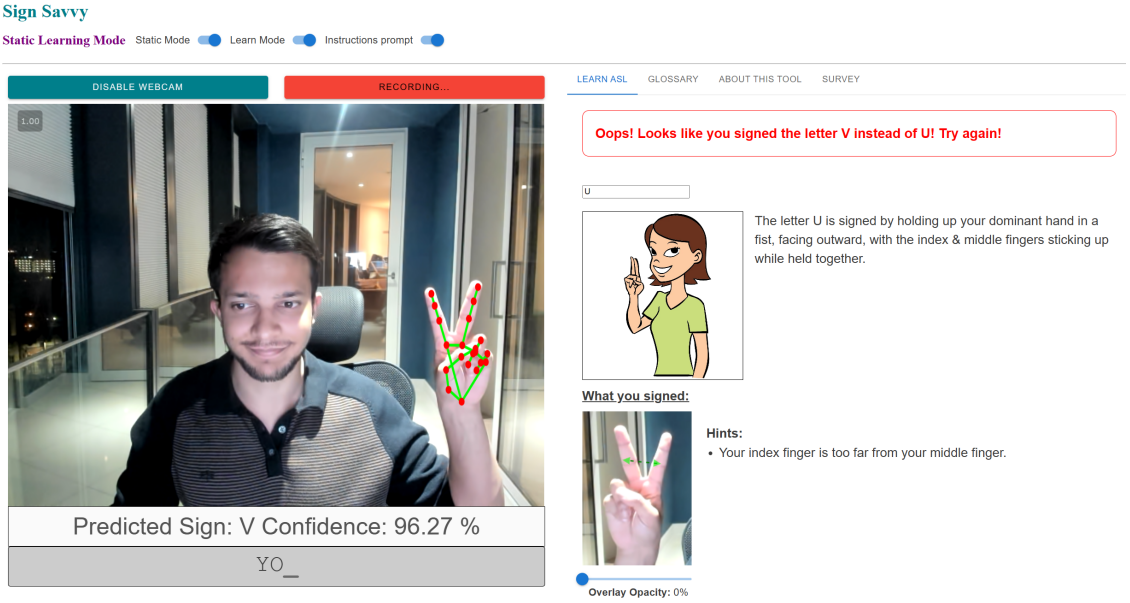


Figure 5.2: Screenshot of SignSavvy's final user interface in static mode

(as shown in Figure 5.2), providing visual cues to indicate the progress of the recording process. To ensure a consistent and modern design, SignSavvy incorporates pre-built components from the Material-UI framework, a popular React component library that follows the principles of Google's Material Design.

Instructional Design and User Guidance: To assist users in understanding and navigating the SignSavvy platform, clear instructions and user guidance are incorporated into the UI. The instructions are adjusted dynamically depending on the specific needs of the user. For example, Our guidance box (shown as the red box in the top right of Figure 5.2) adds instructions on how to start recording in the case where a user has not already done so. We also add an "About This Tool" tab to provide a brief introduction to SignSavvy, its purpose and how to use the tool.

Navigation and Tabs: SignSavvy allows the user to switch between static and dynamic modes using a toggle switch located above the recording component. In static mode, users can focus on learning ASL letter fingerspelling, whilst in dynamic mode, users can learn how to sign words through motion. We also present users with a toggle for "Learn Mode". When learn mode is enabled, the user is guided through signing specific words with demonstration and feedback. When learn mode is disabled, the user can freely fingerspell any word they want, with the output appearing in the type box below the predicted sign box. The information section of the application is tabbed to provide users with easy access to specific sections of the application, with a clear and organised structure for categorising content. The tabbed approach helps users not feel overwhelmed by excessive information and promotes a more focused and streamlined learning experience.

Designing for Accessibility: Accessibility is a fundamental aspect of SignSavvy to ensure that individuals with different abilities can effectively use the application. The UI is designed with clear and contrasting colours to enhance visibility for users with visual impairments. Font sizes and styles are also carefully selected to improve readability. To further enhance accessibility, alternative text descriptions are provided for images and visual elements, enabling screen readers to convey the information to visually impaired users. The UI also implements proper semantic HTML tags to improve compatibility with assistive technologies, as recommended by the WCAG (Web Content Accessibility Guidelines) standard.

5.1.3 ASL Demonstration

The demonstration of ASL signs is a fundamental component of SignSavvy's teaching process, providing users with visual examples and guidance for learning ASL. SignSavvy offers separate

dynamic and static modes to cater to learning aspects of ASL learning. The dynamic mode focuses on teaching users how to sign common ASL words and phrases, whilst the static mode concentrates on teaching users to fingerspell letters, an essential skill in ASL (as discussed earlier in Section 3.2.1). To demonstrate how to perform a sign, SignSavvy integrates various multimedia elements:

- In dynamic mode, users can watch instructional videos of a professional signer demonstrating the proper signing of words and phrases, as shown in Figure 5.3. These videos provide visual cues for users to follow along and mimic the signs accurately. We provide instructions and support for the 12 signs mentioned in Section 4.2.1, chosen after an analysis of popularly used words in everyday conversation, considering input from different ASL resources and communities. It also aims to showcase the system's support for a variety of different hand and pose configurations.

- In static mode, we use a cartoon image (shown in Figure 5.2) to showcase the individual ASL letters, allowing users to focus on the specific hand shapes required for fingerspelling. Each letter is accompanied by a text description that provides instructions on how to perform the sign correctly, along with common pitfalls and mistakes to avoid. Users are instructed to spell out whole words, letter-by-letter, in learn mode. The set of words is chosen such that users can gain exposure to every letter of the alphabet, allowing for ASL fingerspelling proficiency through the application.

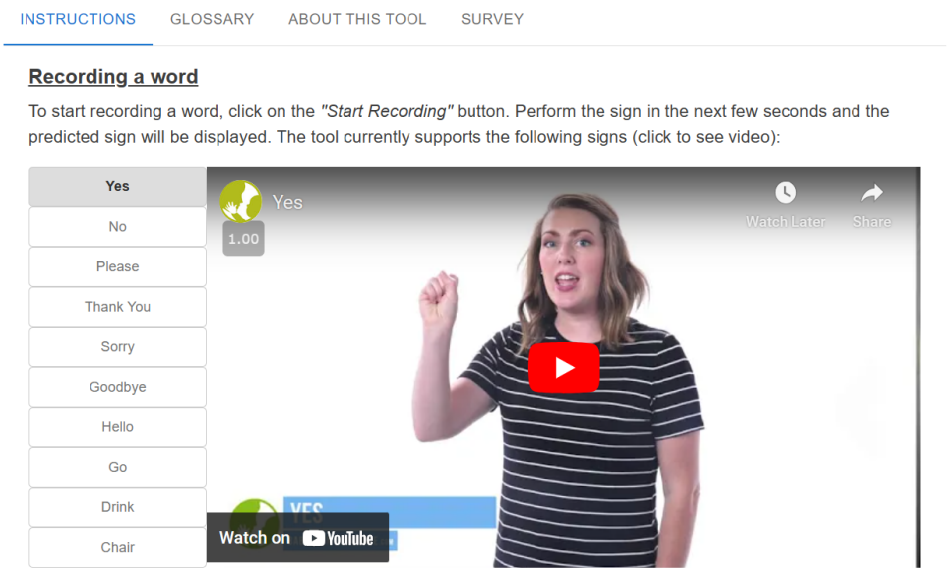


Figure 5.3: Screenshot of dynamic mode video resources demonstrating ASL words [66]

Sign Glossary Section: SignSavvy also features a glossary section, accessible as a tab in the application. This tab serves as a centralised and easily accessible reference for ASL signs, including a comprehensive collection of all available letters and words in the respective static and dynamic modes. The glossary section allows learners to explore the full potential of the system and supplements the main learning mode, which offers a more guided learning experience.

Resource Acquisition: To ensure accuracy and authenticity in the demonstration of ASL signs, SignSavvy uses reputable resources and references. We collaborate with Baby Sign Language [66], a trusted platform widely recognised for its quality ASL content. Through receiving permission for usage of Baby Sign Language's video, image and sign description resources, SignSavvy integrates professionally produced ASL demonstrations, ensuring a reliable and accurate representation of signs for users.

With our design, SignSavvy closely replicates the demonstration experience of a starter course in ASL or real-life tutoring, with both fingerspelling and signing basic words being a crucial first

step to gaining ASL proficiency. SignSavvy leverages the strengths of the existing sign language education infrastructure but also goes a step further, providing an interactive experience and feedback.

5.1.4 Iterative Design and Early Human Evaluation

To ensure a user-centred approach and improve the overall user experience of SignSavvy, an iterative design process based on human-centred design (HCD) principles was followed. Specifically, we followed the methodologies of a study [67] that adopts the ISO 13407 standard for usability. We define our primary application stakeholders to be individuals that have an interest in learning ASL but have little to no experience with signing.

An intermediate evaluation was conducted to gather feedback from users and assess the effectiveness of the platform, motivating changes in later iterations. Specifically, we carried out the evaluation before the implementation of our more detailed geometric analysis of hand and finger orientations (and the corresponding hints), described in Section 5.2.4, and before the implementation of annotation and overlay on the static sign visual playback, described in Section 5.2.5. Dynamic mode was not evaluated due to still being in early-stage development at the time of the intermediate evaluation.

Methodology: A Qualtrics survey was distributed to a diverse group of participants, representing individuals from a variety of backgrounds, all with little to no experience in ASL. The survey aimed to collect quantitative and qualitative feedback regarding their experience with SignSavvy. Participants were first asked basic details about their background in ASL and then asked to answer numerous questions (see Section A.1.1 for specific questions), covering various different aspects of SignSavvy, including:

- UI design and visual appeal
- Ease of use and navigation
- Signs the user found difficult to perform
- System performance and reliability
- Effectiveness and comparison of demonstration methods
- Effectiveness and comparison of ASL teaching/multi-modal feedback
- Open-ended questions and any potential suggestions/requests

In addition to the survey, one-on-one participatory evaluation sessions were conducted with 10 users with no ASL experience. These sessions follow HCD methodologies, allowing for observation, direct interaction, and immediate feedback from users. Users are actively encouraged to explain what they are doing and "think out loud" to generate more accurate insights.

Users' interactions with the application were observed, and they were asked for their feedback on the visual design, layout, features and overall user interface of SignSavvy. Their comments and suggestions regarding the UI were noted to guide design refinements. Users were also asked to perform specific tasks within the application while their interactions were observed. Feedback regarding the ease of performing these tasks and any challenges encountered in the process were collected, helping to identify usability issues and areas for simplification.

Following HCD methodology, we also carried out an expert evaluation with an individual with ASL proficiency to more accurately gauge the performance of the platform demonstration, feedback and classification mechanisms. The expert provides us with more detailed feedback and insights about how well SignSavvy performs as a system, especially when compared to online ASL courses or a real-life tutor. Despite individuals with experience in ASL not being the target audience of the application, this feedback is vital in conjunction with the stakeholder feedback.

Findings and Feedback: In total, there were 22 respondents to the questionnaire and 10 users who participated in an observed user trial of the prototype. The user evaluation yielded several key findings that shape the ongoing development and refinement of SignSavvy:

- **System Responsiveness:** Participants noted that the system was too responsive, registering signs too quickly and not allowing sufficient time to read feedback. This suggests a need to fine-tune the timing of the system's response to create a better user experience. To respond to the feedback, we add strategic timeouts when presenting feedback and demonstrating signs to improve the overall user experience.

- **Cartoon Image vs Real Hand Image**: While the cartoon image demonstrating static ASL signs was generally perceived as friendly, participants expressed a preference for a real hand image in certain instances. They found that a real hand image would be more useful, especially for intricate signs that require precise hand positioning and movement. This feedback highlights the importance of considering alternative visual representations to enhance the accuracy and clarity of sign demonstrations.

- **Detailed Hints:** Participants mentioned that more detailed hints would be beneficial, particularly when they encountered difficulties in performing certain signs. They expressed a desire for clearer guidance on how to correct their hand and finger orientations to improve their signing accuracy. Incorporating more specific and targeted hints could address this issue and enhance the learning experience for users.

- **UI Design and Navigation:** Users generally expressed positive feedback about the UI design and found it visually appealing. However, they provided valuable suggestions for improvement, such as resizing certain elements, aligning elements more consistently, and reducing unnecessary whitespace. These small adjustments can contribute to a more polished and cohesive user interface.

- **Sign Recording Challenges:** Some participants found it challenging to record certain signs, specifically the signs for "V" and "R." They experienced difficulties in capturing the intended hand shapes and movements accurately. This feedback indicates the need for further optimisation of the sign recording functionality to ensure more reliable and accurate sign capture, particularly for complex or intricate signs. While certain signs are expected to be classified less accurately by our static SLR model (see Section 3.4.4), we lower the threshold for the registering of these particular letters from 10 consecutive frames of 70%+ confidence scores to 10 consecutive frames of 50%+ confidence.

- **Usefulness of Confidence Levels:** Surprisingly, participants found the confidence levels provided by the system (see Section 5.2.1) to be highly valuable as feedback. They perceived the confidence levels not only as an indication of the model's output but also as a reflection of their own sign performance. Users appreciated having this additional information, as it helped them gauge their progress and identify areas for improvement. This unexpected observation highlights the psychological impact of confidence levels and suggests their potential as a motivational tool in the learning process.

Furthermore, the expert evaluation provided valuable insights into the overall performance of the system. The expert expressed satisfaction with SignSavvy's usability, interface design, and the effectiveness of the demonstration and feedback mechanisms. The positive feedback from the expert reaffirmed the system's potential as a learning tool for ASL. However, the expert did suggest that the recording system could benefit from some improvements. The expert noted that certain letters required exaggerated hand positions to be recorded, meaning that certain signs could not be performed in a way that is natural to those fluent in ASL. On the other hand, the expert noted that poorly performed versions of signs were often still classified as correct, which could be misleading for a learner who might pick up the wrong technique as a result.

5.2 ASL Teaching Mechanisms

SignSavvy uses a range of teaching mechanisms to enhance the user learning experience and provide effective guidance on how to perform ASL signs more accurately. These mechanisms are designed to offer feedback on the accuracy of the performed signs, deliver clear instructions for improvement and reinforce learning objectives. In this section, we explore the various mechanisms by which SignSavvy provides feedback to the learner on their sign performance and examine how these contribute to the learning experience. Each mechanism plays an important role in providing users with valuable and varied insights, giving them opportunities for guided self-assessment.

5.2.1 Predicted Sign and Confidence Levels

Figure 5.4: SignSavvy predicted sign and confidence level panel underneath recording component

In SignSavvy, the predictions of ASL signs are a key feature that allows users to receive immediate feedback on the correctness of their performed signs. The prediction mechanism differs between dynamic and static modes, using the respective backend infrastructure and cloud-based model.

In dynamic mode, when users record the sign, the aggregated extracted landmarks are sent to the backend for processing. The server forwards the data to the dynamic model, which returns a predicted ASL word based on a pattern similarity algorithm (detailed in Chapter 4).

In static mode, the prediction process is carried out at each frame by reshaping and forwarding extracted landmarks to our feature-based CNN model (detailed in Section 3.4). The output of CNN is a set of probabilities computed by the Softmax function that represents the model's confidence level that a given input corresponds with each possible letter. The letter corresponding to the node with the highest probability value is returned as a predicted sign. The overall probabilities sum to 1, allowing us to easily calculate a confidence percentage of the model's prediction of a given letter. We consider a prediction valid in static mode if the model's output achieves a minimum confidence of 70% or higher. This threshold ensures a reasonable level of confidence for the predicted sign.

The ability for the users to view the predicted sign and associated confidence levels (shown in Figure 5.4) has several benefits. Firstly, it offers transparency and accountability, allowing experienced test users to understand and assess the accuracy of the system's predictions. For those with less experience in ASL, the users can use confidence levels as an additional layer of information, comparing the alignment between their performed sign and the fluctuation of the confidence values. Through this mechanism, users can interactively engage with the platform, receiving real-time feedback on their performance. A simple example of this would be if a user has their fingers too far apart, leading to the model outputting a lower confidence value. Although detailed feedback can correct the user, the learner might instead experiment sooner and find that moving fingers closer together increases confidence values. Our intermediate trial, discussed in Section 5.1.4, showed users put surprisingly high importance on the confidence level value, psychologically treating it as a scoring system and reflection of their own performance rather than the model's confidence in its prediction. One limitation of using model confidence values as indicators of user performance is that it relies on the underlying model being sufficiently accurate and robust.

5.2.2 Interactive Colour-Based Guidance

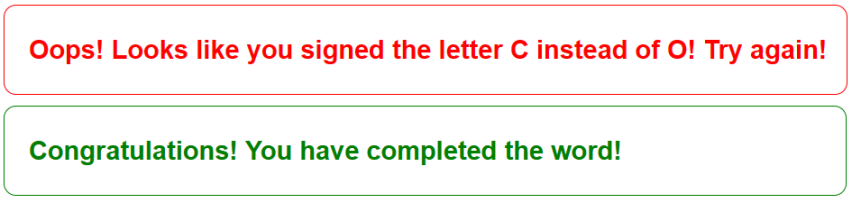


Figure 5.5: Guidance box examples for incorrect signing (above) and correct signing (below)

SignSavvy uses an interactive guidance mechanism to provide users with real-time support and encouragement during the learning experience. This mechanism goes beyond simple text-based feedback, integrating visual cues and messages to guide users through the application. The mechanism involves the use of colour changes, informative messages and friendly encouragement.

The guidance box component contains informative messages to guide users on what signs to perform next. These messages are designed to mimic human-like interactions, delivering instructions and encouragement in a friendly and supportive manner. This allows users to structure their learning and progress systematically through the teaching curriculum of the platform. The messages also guide users on how to navigate the application and resolve technical issues, like providing instructions on how to start recording when the user first starts the app, for example.

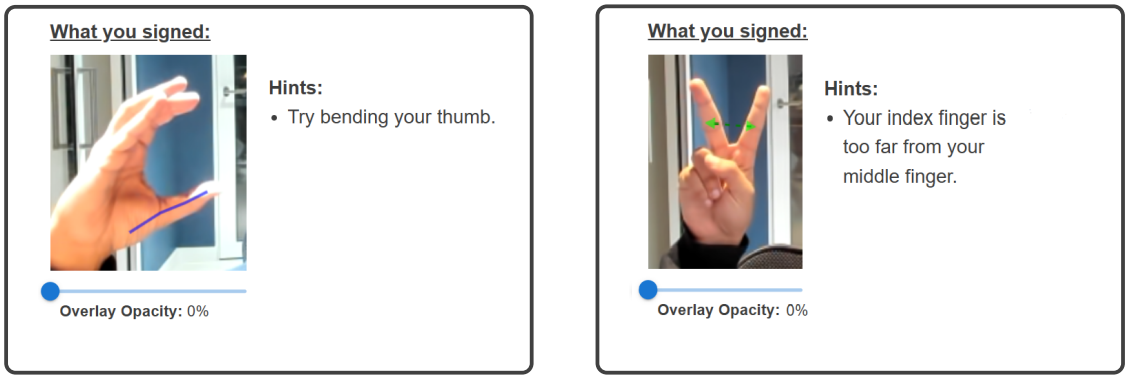
When users perform a sign, the system compares the classified sign with the expected sign that is currently being demonstrated to the user. The guidance box provides immediate visual feedback in the form of a colour-coded indicator. A positive performance is represented by a green colour, indicating that the sign was executed correctly and matched the expected gesture. A red colour, however, indicates a mismatch between the performed sign and the expected gesture. The visual feedback serves as an immediate and intuitive way for users to assess the correctness of their sign and make adjustments accordingly. An example of the indicators is shown in Figure 5.5. Colour-coded indicators are used due to the ability of colour to enhance visual perception and make feedback more immediate and intuitive for users.

Our methodology for the development of our guidance box is based on an abundance of research in the field of education and feedback. One study on the importance of colour as a form of corrective feedback [68] found that indirect colour corrective feedback can be proven to be significant in decreasing students' writing errors, as well as reducing students' anxiety and increasing their level of motivation as compared to other types of feedback. The same study also concluded that students value their own ego and their pride over effective learning; therefore, the types of corrective feedback provided should not be those which make them feel embarrassed or belittled, motivating the introduction of our supportive messages to guide the user.

5.2.3 Visual Playback of Performed Sign

The SignSavvy platform uses visual playback of the sign performed by a user in both static and dynamic modes, each with its own approach and benefits.

Static Mode: In static mode, the visual playback is achieved through the cropping of the user's hand from the webcam stream when a particular letter is registered (after 10 consecutive frames of the same letter classification). By isolating the hand region, SignSavvy creates a focused view of the sign, allowing users to observe their hand shape, orientation and finger positions. The cropping is achieved by defining a bounding box around the hand in the input webcam stream based on the extreme minima and maxima of the extracted MediaPipe landmarks for a frame. Our cropping method is adapted from our end-to-end static SLR approach, with further details explained in Section 3.3.1. The cropped hand image is displayed on a separate crop canvas (as shown in Figure 5.6), through which users can examine their attempt compared to the demonstrated sign and description, identifying any discrepancies and making necessary adjustments.



(a) ASL Letter "C" signed instead of "O"

(b) ASL Letter "V" signed instead of "U"

Figure 5.6: Visual playback of incorrectly performed signs, with annotations and hints as feedback

Dynamic Mode: In dynamic mode, SignSavvy implements video playback of user-performed signs. When a sign recording finishes, the platform saves the recorded video and displays it in a frame object adjacent to the demonstration video. Through this feature, users can observe the fluidity and rhythm of their signing, as well as any inconsistencies or errors in their execution. The recorded video is placed directly next to the demonstration video to allow for direct visual comparison with the ASL professional's signing technique.

Visual playback of performed signs is a powerful learning mechanism that allows users to assess their own signing technique, promoting self-assessment and self-correction. The educational value of visual playback and self-reflection has been discussed in numerous studies [69] [70], highlighting the importance of this feature in SignSavvy for promoting the learning experience for students.

5.2.4 Hand and Finger Orientation Hints

To provide more specific and detailed feedback, SignSavvy carries out a detailed geometric analysis of the user's hand and finger orientations to provide hints on how to improve. This analysis specifically involves calculating the orientation of the hand and the angles of finger bend and finger spacing.

Hand Orientation: The orientation of the hand in 3d space can be described using Euler angles. In the case of hand orientation, the values of pitch, yaw and roll are commonly used, describing the rotation of the hand in different axes, as shown in Figure 5.7. We choose to use Euler angles for their ease of calculation and easy interpretability. However, Euler angles can suffer from issues such as gimbal lock, where a particular combination of rotations can result in a loss of one degree of freedom. Other representations, such as quaternions, may be more robust in some cases [71].

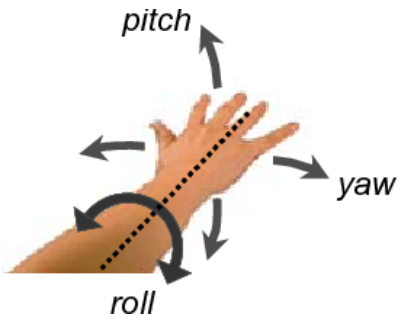


Figure 5.7: Euler angle representation of hand orientation (pitch, yaw and roll) [72]

To calculate the hand orientation, we first define the plane of the palm to represent the hand in 3d space. Specifically, we use the MediaPipe hand landmarks 0, 5 and 17 (from Figure 2.9a) as

the 3 points to define the palm. The algorithm can be summarised as follows:

1. Calculate the vectors $\mathbf{v}_1$ (from landmark 0 to 5) and $\mathbf{v}_2$ (from landmark 0 to 17):

$$\mathbf{v}_1 = \begin{pmatrix} \text{landmark}_5.x - \text{landmark}_0.x \\ \text{landmark}_5.y - \text{landmark}_0.y \\ \text{landmark}_5.z - \text{landmark}_0.z \end{pmatrix}, \mathbf{v}_2 = \begin{pmatrix} \text{landmark}_{17}.x - \text{landmark}_0.x \\ \text{landmark}_{17}.y - \text{landmark}_0.y \\ \text{landmark}_{17}.z - \text{landmark}_0.z \end{pmatrix}.$$

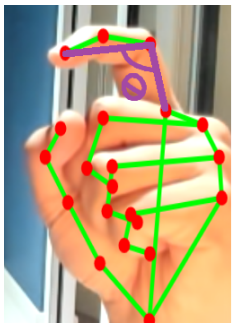
2. Compute the cross product, using the equation $\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$, where $\times$ denotes the cross product operation and $\mathbf{n}$ denotes the normal vector to the plane formed by landmarks 0, 5, and 17. The direction and magnitude of $\mathbf{n}$ provide information about the hand orientation.

3. Calculate the pitch (θ_{pitch}), yaw (θ_{yaw}) and roll (θ_{roll}) angles:

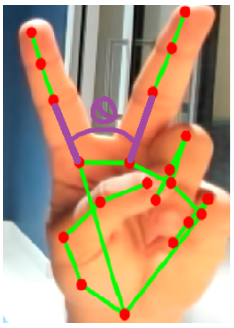
   - $\theta_{\text{pitch}} = \arctan\left(\frac{\mathbf{n}_y}{\mathbf{n}_z}\right)$, where $\mathbf{n}_y$ and $\mathbf{n}_z$ are the y and z components of $\mathbf{n}$ respectively.
   - $\theta_{\text{yaw}} = \arctan\left(\frac{\mathbf{n}_x}{\mathbf{n}_z}\right)$, where $\mathbf{n}_x$ and $\mathbf{n}_z$ are the x and z components of $\mathbf{n}$ respectively.
   - $\theta_{\text{roll}} = \arctan\left(\frac{\mathbf{n}_y}{\mathbf{n}_x}\right)$, where $\mathbf{n}_x$ and $\mathbf{n}_y$ are the x and y components of $\mathbf{n}$ respectively.

Finger Orientation: SignSavvy's finger analysis calculates both the bend of each finger and the spacing between each finger. To calculate angles between parts of the finger, we first define 3-dimensional vectors based on specific finger connections.

- To calculate the bend of the finger, we utilise the MediaPipe landmarks for the MCP (metacarpophalangeal) joints, PIP (proximal interphalangeal) joints, and TIP (fingertip) of each finger. As shown in Figure 5.8a, we define a vector from the MCP (base of the finger) to the PIP (first joint) and another vector from the PIP to the TIP (fingertip). We then calculate the angle between these two vectors using the dot product formula (as featured in Section 4.3.1), obtaining an overall measure of how bent a particular finger is.
- To calculate the spacing between adjacent fingers, we define vectors from the MCP joint to the PIP joint of the finger (as shown in Figure 5.8b, where we define the vectors for the index and middle fingers). We then calculate the angle between the two adjacent fingers' vectors using the dot product formula to obtain a measure of finger spacing.



(a) Index finger bend angle θ



(b) Index and middle finger spacing angle θ

Figure 5.8: Annotations showing relevant vectors and angles (purple) in finger orientation analysis

We carry out hand orientation, finger spacing and finger bend analysis on a subset of 100 of our original training samples from the WLASL dataset [27] from each class, averaging out values to obtain ground truth values for expected orientations. The result of the dataset analysis is stored in a JavaScript object for reference by SignSavvy. When a sign is registered and does not match the expected letter class, we perform the hand and finger orientation analysis on the frontend using React's built-in `Math` library, comparing the values to the expected values. If a calculated

orientation value is sound to be significantly different (beyond a certain threshold), SignSavvy generates intuitive and comprehensible text hints to guide the user on how to improve.

The generated hints are designed to be specific and informative, addressing the feedback we received in our intermediate evaluation, discussed in Section 5.1.4, by providing more in-depth details on corrective actions users can take in addition to rating their performance. For example, in our hand orientation analysis, we present hints such as "Try rotating your wrist more" when the roll value is found to be significantly less than the expected value. An example of finger analysis can be seen in Figure 5.6a, the user signs the ASL letter "C" instead of "O", resulting in the thumb value having a much lower bend angle than expected and the generation of feedback to "try bending the thumb". Similarly, Figure 5.6b showcases an example of finger spacing analysis between the index and middle finger being much larger than expected when "V" is signed instead of "U". In both cases, the offending fingers are specifically mentioned by name.

For dynamic mode, we perform a simplified analysis of only finger bend, skipping finger spacing and hand orientation calculations. The finger bend analysis is performed for the frames corresponding to quarter way (start), mid-way (middle) and three-quarters way (end) of the recorded sign and is compared to the average finger bend values averaged out over the dataset videos at the same time stages to generate feedback hints.

5.2.5 Image Annotation and Overlay for Static Signs

In addition to the visual playback of static signs, as discussed in Section 5.2.3, SignSavvy enhances the visual feedback by annotating the cropped image of the user's performed sign. Specifically, we annotate problematic fingers with a blue line over the finger for finger bend issues (as shown in Figure 5.6a) and a green arrow between two fingers for finger spacing issues (as shown in Figure 5.6b). By specifically highlighting problem areas, users are provided with specific finger adjustments in a visual manner that is more interpretable than just text hints alone. Our choice to feature both visual and text feedback is motivated by dual coding theory, which concludes that combining visual representations with verbal instructions improves learning outcomes [70].

SignSavvy also offers an image overlay feature for static mode playback. This feature overlaps a cropped hand image of the correct hand configuration of an ASL letter over the user's performed sign playback. The overlay is adjustable using an opacity slider, allowing users to control the transparency of the overlayed image. An example is shown in Figure 5.9, where the letter "C" is signed instead of "O". At 50% opacity, the overlay shows clearly the area of overlap between the performed and expected sign, but also clearly highlights the need for the hand to be more rounded to achieve a correct classification.

The overlay feature provides users with a visual reference for ideal hand configuration. By superimposing the professionally performed image over their own attempt, users can easily compare their hand shape and finger positions to the desired standard. A real image of the hand was used in the overlay to address the feedback of the intermediate human trial (as discussed in Section 5.1.4), where users expressed that the cartoon image used for demonstration lacked clarity and made a comparison to the playback difficult in certain scenarios.

Figure 5.9: User performed sign, overlayed by target letter "O" with opacity 0%, 50% and 100%

Chapter 6

Evaluation

The evaluation of an interactive learning system is a critical step in assessing its effectiveness in teaching, usability and overall performance. In this chapter, we present the evaluation of the final SignSavvy platform, focusing on user trials and experiments conducted to gather feedback from a diverse range of backgrounds. We also carry out a technical evaluation of the system performance, assessing the scalability and robustness of the final integrated system.

We have carried out detailed evaluations of our final SLR pipelines in earlier chapters, namely Section 3.4.4 for our feature-based static pipeline and Section 4.4 for our dynamic pipeline. In these evaluations, we analyse the classification accuracy of the developed models on various different datasets and in real-life usage when integrated with a full user-recording pipeline. We also analyse specific strengths and weaknesses of our final models, performing a more detailed analysis of the classification strengths, scalability and efficiency of the models. From this, we identify possible methods for improvement in the future and discuss the suitability of our models for our requirements relative to the state-of-the-art.

We also carry out an intermediate evaluation (see Section 5.1.4) involving the use of a questionnaire, one-on-one observed user trials and an expert evaluation. This earlier evaluation serves to allow for the iterative development of the platform, resulting in the final version that is discussed further in this chapter. Many of our more advanced features, such as geometric analysis and image overlays, as well as numerous user interface and experience changes, are motivated by the feedback of this evaluation.

In the following sections of this chapter, we present the methodology employed for our final user trials, including the participants involved, tasks assigned, and metrics used to measure usability and performance. We also present the results of the evaluations, highlighting key findings, user feedback and an analysis of the system's effectiveness at interactively teaching ASL. We also carry out a brief technical evaluation of the final developed SignSavvy platform.

6.1 Final User Evaluation

6.1.1 User Survey

In order to gather feedback on the usability, user satisfaction and effectiveness of our final iteration of the SignSavvy platform, we conducted a survey among participants who used the platform. As discussed in literature related to human-centred design (HCD) methodologies [67], surveys are an effective way to gather feedback from a larger number of users and obtain both qualitative and quantitative data. As per survey recommendations, our survey is designed to ask a mix of open and closed questions, capturing user perceptions for different components of the SignSavvy platform, satisfaction levels, and suggestions for improvements. From these insights, we can inform future enhancements and evaluate the strengths and weaknesses of the current final system.

We design our survey with careful consideration, motivated by existing literature on effective methods of gathering feedback. Some key design decisions are summarised as follows:

1. **Likert Scale:** A Likert scale [73] was used for participants to rate their agreement or satisfaction level on a scale ranging from 1 to 5. This scale allowed for a structured and quantitative assessment of participants' opinions and perceptions. Our specific rating scale ranges from poor to excellent, with poor=1, average=2, fair=3, good=4 and excellent=5.

2. **Adapting Questions from popular usability surveys:** To ensure the reliability and validity of our survey, we incorporated select questions from established usability surveys. In particular, we adapted questions from version 3 of the PSSUQ (Post-Study System Usability Questionnaire) [74]. By adapting existing surveys, we were able to include questions that have been extensively tested and validated, allowing for more accurate and comparable measurements of user satisfaction and perceived usability.

3. **Clear and Concise Language:** The survey questions were formulated using clear and concise language to minimise ambiguity. We avoided technical jargon and used simple terms to ensure that the questions were accessible and easy to comprehend for users with varying levels of familiarity with technical and system-related terms.

4. **Mixed Methodology:** The survey included a combination of closed-ended and open-ended questions to capture both quantitative and qualitative feedback. Closed-ended questions with Likert scale ratings allowed for quantitative analysis and provided measurable data on user perceptions. Open-ended questions encouraged participants to provide detailed feedback, suggestions, and insights, offering qualitative data to gain deeper insights into their experiences and uncover potential areas of improvement.

5. **Testing Different Areas of the System:** The survey covered various different aspects of the SignSavvy platform to gather feedback on different components and functionalities. Participants were asked to evaluate the effectiveness of sign demonstrations, the usefulness of the teaching mechanisms, the user interface (UI) and its ease of use, and overall user satisfaction. By assessing multiple dimensions of the system, we aim to form a comprehensive understanding of user perceptions across different areas and identify specific strengths and areas for improvement within the platform.

We deployed the survey using the Qualtrics online survey platform, allowing for easy data collection and analysis. Specifically, we reuse some of the same survey questions from our intermediate evaluation (Section 5.1.4) to allow for direct comparison of results and measure improvement of our platform over iterations. The survey was active for a duration of one week, allowing participants sufficient time to provide their feedback on the SignSavvy platform. To ensure easy access for participants, we included a survey tab within the SignSavvy application (shown in Figure 5.2), providing a direct link to the survey. The survey link was also directly shared on social media. Some participants had never previously used the SignSavvy platform, whilst others had participated in the intermediate evaluation survey or user observations.

In total, the survey received 23 responses from participants who used the SignSavvy platform. To gain meaningful insights and facilitate the analysis of the feedback, we categorise the survey results and subsequent discussion into three main categories: user interface and ease of use, system performance and reliability, and ASL demonstration and teaching mechanisms.

Each category consists of five questions, corresponding to various relevant components, that are rated from 1-5 using the Likert scale described previously. For each component, participants are able to optionally add additional detail of what they felt did not work as well and any suggestions they might have, allowing us to also generate qualitative feedback from the survey. Where candidates provided a score of 3 or less for a specific component, they are specifically prompted to suggest any additional feedback. Specific questions, with more detailed descriptions of components, can be found in Section A.1.2.

User interface and ease of use: This category aimed to assess participants' thoughts on the SignSavvy platform's user interface (UI) and its overall ease of use. Participants were asked to provide feedback on various aspects of the UI design and navigation.

Component	Average Score (1-5)
Clarity of the user interface	3.87
Intuitiveness of tab layout	4.22
Ease of accessing learning materials	4.30
Visual appeal of the platform	4.17
Consistency of design elements	3.96
Overall UI and ease of use	4.10

Table 6.1: User Interface and Ease of Use - Average Survey Scores

Comparing the results from Table 6.1 to the findings from our intermediate survey, we observed notable improvements in the visual appeal of the platform (with the same question used across both surveys), which saw a substantial increase in the average score from 3.42 to 4.17. Additionally, the intuitiveness of the tab layout received a high average score of 4.22, indicating that participants found it easy to navigate through the different sections of the platform.

While the average scores indicate positive feedback, some participants mentioned the need for clearer labelling and guidance, particularly regarding the purpose of the "glossary" tab and the difference between the static and dynamic modes, where users noted it was unclear what these features involved without clicking onto them. This feedback highlights areas for improvement in terms of providing more explicit instructions and clarifying the functionality of certain features. Overall, SignSavvy achieves an overall average score of 4.1 out of 5 for its UI, suggesting significant progress in this area and validating our efforts made to enhance the visual appeal, navigation and accessibility of the platform.

System Performance and Reliability: This category aimed to evaluate participants' thoughts on the performance and reliability of the SignSavvy platform. Participants were asked to provide feedback on various aspects related to system responsiveness, stability, and technical issues.

Component	Average Score (1-5)
Responsiveness of the platform	4.26
Loading speed of content	4.13
Stability of the platform	3.91
Technical issues encountered	3.43
Accuracy of sign recognition	3.65
Overall system performance and reliability	3.88

Table 6.2: System Performance and Reliability - Average Survey Scores

From Table 6.2, we observe that participants reported a high level of satisfaction with the platform's responsiveness and content loading speeds. Many participants cited the speed of the static mode detection to be particularly impressive, with other users who participated in the intermediate evaluation commenting on noticeable improvements in the overall pacing of the application.

However, participants identified areas for improvement, particularly regarding platform stability and technical issues, with an average score of 3.91 and 3.43, respectively. Some users encountered technical issues, such as the system freezing or the MediaPipe landmark detection failing when switching between pipelines. Additionally, many users continued to face some issues with the accuracy of the sign recognition, especially in dynamic mode, where users occasionally received

the wrong classifications even after following the instructional videos accurately. Although static recognition was generally accurate, some users faced challenges in performing certain letter signs, particularly the letters "U" and "T." This feedback further emphasises our findings in Section 3.4.4, highlighting the need for additional improvements to make the MediaPipe landmark extraction and classification CNN model more robust.

Overall, the average score of 3.88 for system performance indicates a satisfactory result for SignSavvy. Given the limited development time, the feedback suggests that the platform is usable and responsive. However, it also emphasises the importance of further debugging, testing, and optimisation to enhance overall system stability, ensuring that technical issues do not hinder the overall ASL learning experience of the user.

ASL Demonstration and Teaching Mechanisms: This category aimed to evaluate participants' thoughts on the various ASL demonstration methods and teaching mechanisms used in the SignSavvy platform. Participants were asked to provide feedback on the effectiveness and usefulness of these instructional components.

Component	**Average Score (1-5)**
Engagement and enjoyment	4.35
Overall learning effectiveness	4.35
Demonstration effectiveness	3.91
Teaching mechanisms effectiveness	4.74
Overall ASL demonstration and teaching mechanisms	4.34

Table 6.3: ASL Demonstration and Teaching Mechanisms - Average Survey Scores

These results indicate that participants highly valued the ASL demonstration and teaching mechanisms integrated into the SignSavvy platform. The average scores in Table 6.3 show a positive perception of these instructional components. Participants reported high levels of engagement and enjoyment, as well as noting they feel the platform has a strong impact on helping to learn ASL effectively. The teaching mechanisms, in particular, received a significantly high average score of 4.74, indicating their usefulness in facilitating the learning process. Many users commented on the interactive game-like nature of the platform, helping keep their interest throughout their evaluation. A lower average score of 3.91 was given to demonstration effectiveness, with some users mentioning that sign descriptions are too long and that they spent very little time focusing on the demonstrations before attempting a sign themselves. This highlights the need to integrate our demonstrations and our teaching mechanisms into a more cohesive and intertwined user experience, making sure demonstrations are visual and succinct.

In addition to the Likert-scale-based responses, participants were asked to rank the different teaching mechanisms from most useful to least useful, as well as provide reasoning for their decision. The aggregated rankings based on participant responses are as follows: 1) Visual playback of performed sign, 2) Image annotation and overlay, 3) Predicted sign and confidence levels, 4) Hand and finger orientation hints, and 5) Interactive colour-based guidance.

The rankings highlight the importance of visual aids and comparisons in the learning experience, with participants finding the visual playback of signs and image annotations/overlays particularly beneficial, especially in static mode. The combination of being able to observe the sign execution and assess and compare hand configuration with the adjustable overlay was praised by many participants. As noted in the intermediate evaluation, many users also found confidence levels a useful indicator of their own performance to allow for real-time adjustment.

The lower ranking of the orientation hints is a surprising result given their more fine-grained and specific nature. However, some users noted that the hints were sometimes inaccurate or not provided when needed, which may have affected their perceived usefulness. This feedback suggests the importance of improving our calculation process and tuning thresholds for presenting hints. It also suggests more types of hints could help users more specifically target the issues in their

signing technique beyond hand orientation and finger spacing/bend.

Overall, SignSavvy's ASL demonstration and teaching mechanisms achieved an impressive average score of 4.34 out of 5, indicating their effectiveness in supporting ASL learning. The positive feedback received emphasises the significance of incorporating visual aids and interactive elements into the platform to enhance the overall learning experience for users.

6.1.2 Comparative Study

To assess the performance and effectiveness of SignSavvy in comparison to other existing ASL learning tools, we conducted a comparative study. The objective of this study was to evaluate the learning outcomes and user experience of SignSavvy in comparison to a traditional online learning approach, based on using only instructional content such as videos, photos, and textual descriptions of ASL signs. Through our controlled experiments, we gain an insight into the unique advantages and potential areas for improvement of the SignSavvy platform.

Methodology: We recruited a total of 8 participants with no prior experience in ASL or the SignSavvy platform. The participants ranged in academic background, nationality, and spoken language proficiency. The participants were randomly divided into two groups: the SignSavvy group (Group A) and the traditional learning group (Group B). Each group consisted of four participants.

Group A (SignSavvy group) learned a set of 20 ASL letters and 5 ASL words using the SignSavvy platform (using both static and dynamic mode respectively). They received interactive sign demonstrations, feedback, and hints provided by the system. The set of letters to be learned was presented to the users in a round-robin fashion, asking the user to perform each letter before repeating the whole set. Group B (traditional learning group) learned the same set of words and letters through a combination of instructional videos, photos, and textual descriptions. They were encouraged to study and practice using these traditional materials without a guided learning experience but with access to all the resources they need. Both groups were given a time frame of 30 minutes to try to learn and practice the 10 letters and 3 words to their best effort.

To evaluate the learning outcomes, an ASL-accredited individual assessed the participants' signing performances. The ASL expert asked each participant to demonstrate their understanding and ability to perform the learned signs. Specifically, the participants were assessed on a standardised subset of 5 ASL letters and 3 ASL words from the overall learned signs. The participants were evaluated on their accuracy, fluency, and overall proficiency. The ASL expert rated each participant's attempt on a scale of 1 to 5 for each letter and word, with 1 being poor and 5 being excellent. The individual scores were aggregated to obtain a final performance score for each participant. If the user forgot the sign entirely, a score of 1 was given, with 5 indicating near-perfect execution of a sign.

Participant	Group	Static Letters Score (/25)	Dynamic Words Score (/15)	Overall Score (/40)
1	A	20	11	31
2	A	18	9	27
3	A	19	9	28
4	A	16	10	26
5	B	11	8	19
6	B	16	10	26
7	B	14	8	22
8	B	15	9	24

Table 6.4: Summary of comparative study results

Results and findings: The comparative study revealed significant differences in learning outcomes between the SignSavvy group (Group A) and the traditional learning group (Group B). Group A, which used the SignSavvy platform, achieved higher scores in both the static letters and

dynamic words assessments compared to Group B, who relied on traditional learning methods. Results for each participant are shown in Table 6.4.

The SignSavvy group demonstrated a mean static letters score of 18.25 out of 25, while the traditional learning group achieved a lower mean score of 14.0 out of 25. Similarly, for the dynamic words assessment, the SignSavvy group attained a mean score of 9.75 out of 15, whereas the traditional learning group scored an average of 8.75 out of 15. When considering the overall score combining both static letters and dynamic words, the SignSavvy group had a mean score of 28.0 out of 40, while the traditional learning group achieved a lower mean score of 22.75 out of 40.

Statistical Analysis: To determine the statistical significance of the differences in mean scores between the SignSavvy group (Group A) and the traditional learning group (Group B), a two-sample, one-tailed, independent t-test was conducted at a 5% significance level ($\alpha = 0.05$) on the overall scores. The t-test assesses whether the observed differences in mean scores are statistically significant or merely due to random chance. We define a null hypothesis that SignSavvy provides no benefit to learning ASL signs compared to traditional methods and propose an alternative hypothesis that there are improvements in learning outcomes by using SignSavvy. Our statistical test allows us to conclude that there is a significant improvement in learning outcomes when using the SignSavvy platform (M=28.0, SD=2.16) compared to traditional methods of online ASL learning (M=22.75, SD=2.99), $t(6) = 2.85$, $p = .146$. As $p < 0.05$, we can establish a statistical significance from our test at the 5% significance level. This result is significant in validating the importance of a feedback-based approach to sign language that motivates our project.

Expert Feedback: After the experiment, we collected feedback from our expert to gain a more in-depth insight into the strengths and weaknesses observed in the groups' learning methods. Over a larger set of 20 ASL letters, participants in the traditional learning group had a higher tendency to forget signs altogether, while participants in the SignSavvy group demonstrated better retention. The expert observed that the discrepancy in scores between the two groups often resulted from differences in retention and memory rather than significantly more accurate sign performances. These findings suggest that SignSavvy's current advantage lies in promoting higher retention and engagement, while further development of the teaching mechanisms is needed to enhance the actual performance of sign elements such as hand shapes and configurations.

Participant Feedback: Qualitative feedback from participants further emphasised the benefits of the SignSavvy platform. Participants in the SignSavvy group found the hints and confidence levels particularly useful in learning how to correctly perform a sign. They appreciated the guided nature of the learning process and enjoyed using the platform, finding it engaging throughout the 30-minute session. In contrast, participants in the traditional learning group reported feeling lost and lacking direction when relying solely on resources. The absence of interactive elements led to boredom, as they had to repeatedly review static images of signs. These insights underscore the intuitive interface and engaging learning activities provided by SignSavvy.

Limitations of our study: It is essential to acknowledge the limitations of the comparative study. The short time frame and small set of signs may not accurately represent the long-term retention and performance of newly learned signs. Additionally, the evaluation relied on the judgement of a single ASL expert and involved a limited number of participants. Conducting a more extensive and diverse study with a larger sample size would provide more comprehensive insights into the effectiveness of SignSavvy and traditional learning methods for ASL.

Conclusion: The findings of the comparative study highlight the distinct advantages of SignSavvy over traditional learning approaches for ASL. The interactive nature of the platform, combined with real-time feedback, hints, and guided visual demonstrations, contributed to improved learning outcomes and enhanced user experience. However, further improvements to teaching mechanisms are necessary to enhance not only the retention of new signs but also the overall signing technique. These findings reinforce the value of incorporating technology-driven solutions in ASL education and provide a foundation for ongoing enhancements and refinements of the SignSavvy platform.

6.2 Technical Evaluation

In addition to user testing and accuracy assessments, a comprehensive technical evaluation was conducted to measure the performance and latency of the SignSavvy system. We used the Chrome Developer Tools' Performance tab to measure various latency metrics, with the results shown in Table 6.5. Multiple measurements were taken by recording particular tasks, with the results averaged over 3 runs to provide a more accurate understanding of the system's performance. Performance measurements were conducted on a machine with the following specifications: *Windows 10, 11th Gen Intel Core i7-1165G7 Processor @ 2.80GHz, 32GB RAM*. The actual performance of SignSavvy may vary depending on the hardware and network conditions of the user's device.

Measurement	**Average Latency (seconds)**
Original Page Load	1.78
Static Model Download (No Cache)	5.42
Static Model Download (With Cache)	1.24
Sign Classification (Static)	0.32
Sign Classification (Dynamic)	3.41
Glossary Search Response	0.48
Video Playback	0.96

Table 6.5: Average latency measurements from the Chrome Developer Tools profiler

The latency measurements provide insights into the performance and responsiveness of the SignSavvy system. The average page load time of 1.78 seconds indicates a relatively fast initial loading of the platform. The static model download time without cache (5.42 seconds) and with cache (1.24 seconds) highlights the benefit of using caching for efficient serving of pre-downloaded content, reducing download time after the initial visit.

The sign classification latency for static signs is impressively low, with an average of 0.32 seconds. This indicates a swift recognition process, enabling real-time feedback for users performing static signs. However, the dynamic sign classification latency is considerably higher, averaging at 3.41 seconds. The increased latency in dynamic sign classification can be attributed to the complexity of the backend communication and similarity comparisons.

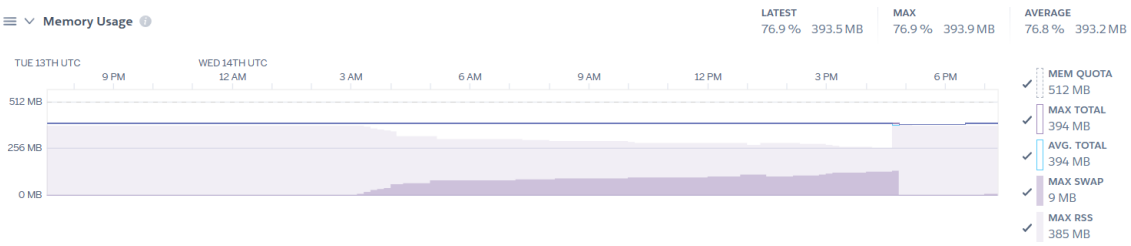


Figure 6.1: Server memory usage for 3 concurrent users, from Heroku metrics tool

Scaling tests were also conducted to assess the server's performance under increased load. During the tests, the memory usage for three concurrent users peaked at 394MB out of the 512MB Heroku quota, as shown in Figure 6.1. This indicates that the system operates within the allocated memory limits. However, for further scalability, upgrading the Heroku dyno to a plan with more RAM would allow for accommodating a larger number of concurrent users without resource constraints.

Continuous monitoring and optimisation are essential to maintain and improve the performance of SignSavvy in the future. Ongoing optimisation efforts allow for the identification of potential bottlenecks in our architecture and design, allowing us to make modifications to enhance the overall user and learning experience.

Chapter 7

Conclusion

This research project has successfully achieved its objectives in exploring and developing static and dynamic Sign Language Recognition (SLR) pipelines and constructing the SignSavvy platform, to provide an immersive learning experience for ASL learners. The project's conclusions summarise the key insights gained throughout the study.

Our exploration of static SLR pipelines involved investigating various architectures, leading to the development of a high-performing lightweight feature-based model. This model demonstrates comparable classification results to state-of-the-art methods, achieving an accuracy rate of 90.3% on the test set and an impressive 88.5% when integrated within a real-world SLR pipeline. It is optimised for real-time usage in interactive learning environments, with an average latency of just 0.22 seconds for sign classification. Through the novel use of the MediaPipe library [6] to obtain localised hand features, we have enhanced the accuracy and performance of our feature-based approach compared to other architectures like an end-to-end approach based directly on images.

For dynamic sign recognition of signs that require motion, we developed a custom pipeline that captures and analyses the temporal aspects of sign language, using a similarity-based approach for classification. The dynamic pipeline achieves an accuracy rate of 86.7% on our test data, showing its effectiveness in recognising and interpreting dynamic signs. Our developed model avoids the need for intensive deep learning models that are often used in dynamic SLR studies, achieving an average latency of just 1.32 seconds for classification, providing a light-weight solution to dynamic SLR for real-time applications.

The integration of the developed static and dynamic SLR pipelines into the SignSavvy platform has resulted in a comprehensive learning experience for ASL learners. SignSavvy offers an immersive environment that combines pipeline architectures and provides a user-friendly interface for interactive learning and practice. The platform is designed to be easily accessible from multiple devices in the form of a web app, without the need for any specialised recording equipment. By integrating curated ASL sign demonstrations and exploring various design and architecture decisions, we have created an accessible, robust, and reactive platform that allows for efficient sign language learning.

To enhance the learning process for sign language learners, SignSavvy implements feedback mechanisms that deliver accurate and personalised feedback to signers, providing targeted suggestions for improvement and simulating the guidance of a real-life tutor. The addition of comprehensive feedback on sign performance further enhances the learning process and sign proficiency of the users. We have developed a range of novel multi-modal feedback mechanisms, ranging from visual cues and annotations to detailed geometric analysis of hand and finger orientations.

The evaluation results obtained through comparative studies and survey analysis confirmed the significance of a feedback-based approach in sign language learning. The statistical analysis highlighted the importance of incorporating feedback mechanisms in the learning process, reinforcing the effectiveness and impact of SignSavvy. The survey results reinforced the positive user experience and satisfaction with SignSavvy, emphasising its value in supporting ASL learning, while

providing insights into areas of improvement and guiding the future direction of the platform.

In conclusion, this research project contributes to the advancement of the fields of sign language recognition and learning. By combining the developed SLR pipelines with the innovative SignSavvy platform, we have created a promising avenue for individuals to learn and communicate through sign language. This achievement not only promotes inclusivity but also ensures that learning sign language is accessible and engaging for all.

7.1 Future Work

While we have made significant strides in the development of sign language recognition pipelines and the creation of the SignSavvy platform, there is a huge scope for further development to improve the effectiveness of interactively teaching ASL:

1. **Combining similarity-based dynamic SLR with deep learning models**: The custom dynamic pipeline developed in this research can be improved by investigating the combination of our similarity-based approach with advanced deep learning models like Hidden Markov Models, leading to improved overall temporal modelling.
2. **Integrating facial landmarks for improved non-manual signal detection**: Although our models incorporate pose and hand landmarks, non-manual signals (such as eyebrows, eyes and lips) play an important role in ASL communication. Integrating fine-grained MediaPipe Face landmarks would improve the robustness of our pipelines.
3. **Improving the reliability and features of the SignSavvy platform**: Improving the reliability and robustness of the SignSavvy platform is crucial for its widespread adoption. Optimising performance, addressing potential technical challenges, and refining the user interface and experience can contribute to creating a more seamless and user-friendly platform.
4. **Expanding sign language support and vocabulary**: SignSavvy can be extended to support a broader range of signs and gestures, including non-letter static signs and more dynamic words. Expanding the sign language vocabulary will increase the utility of SignSavvy and provide users with a more comprehensive learning experience.
5. **Combining static and dynamic modes into one cohesive experience**: Integrating static and dynamic sign language recognition pipelines within SignSavvy can provide learners with a comprehensive approach to sign language education. This integration can involve seamlessly transitioning between static and dynamic signs, providing simultaneous feedback and evaluation on both types of signing, and ensuring a consistent, intuitive and immersive learning environment for users. This integration could be expanded further with full ASL sentences, allowing users to learn about ASL language grammar and gain true proficiency.
6. **More detailed and specific geometric analysis and hints**: A more precise and specific geometric analysis of hand and finger orientations, along with the added analysis of factors such as pose and facial expressions, would help learners improve their technique and produce more accurate and expressive signs.
7. **Creating a fully guided learning experience**: Developing a fully guided learning experience within SignSavvy, including structured lessons, curriculum-based progression, and interactive or game-like exercises can provide users with a comprehensive, engaging and systematic approach to learning sign language.

Bibliography

[1] World Federation Of The Deaf (WFD). Human rights of the deaf, Aug 2017. URL https://wfdeaf.org/our-work/human-rights-of-the-deaf/. [Accessed 24-Jan-2023].

[2] Duolingo. URL https://www.duolingo.com/. [Accessed 08-Jun-2023].

[3] Susan Daniels OBE. The lack of sign language in schools is an injustice for young people, deaf and hearing, May 2017. URL https://www.huffingtonpost.co.uk/susan-daniels-obe/the-lack-of-sign-language_b_16623174.html. [Accessed 08-Jun-2023].

[4] W.H.Organization. Deafness and hearing loss, Apr 2021. URL https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss. [Accessed 24-Jan-2023].

[5] C. Valli and C. Lucas. *Linguistics of American Sign Language: An Introduction*. Clerc books. Gallaudet University Press, 2000. ISBN 9781563680977. URL https://books.google.co.uk/books?id=mfS3GlTLAUMC.

[6] Mediapipe solutions guide. URL https://developers.google.com/mediapipe/solutions/guide. [Accessed 30-May-2023].

[7] Danielle Bragg, Oscar Koller, Naomi Caselli, and William Thies. Exploring collection of sign language datasets: Privacy, participation, and model performance. 2020. doi: 10.1145/3373625.3417024. URL https://doi.org/10.1145/3373625.3417024.

[8] Timnit Gebru. Race and Gender. In *The Oxford Handbook of Ethics of AI*. Oxford University Press, 07 2020. ISBN 9780190067397. doi: 10.1093/oxfordhb/9780190067397.013.16. URL https://doi.org/10.1093/oxfordhb/9780190067397.013.16.

[9] Rose Stamp. Do Signers Understand Regional Varieties of a Sign Language? A Lexical Recognition Experiment. *The Journal of Deaf Studies and Deaf Education*, 21(1):83–93, 09 2015. ISSN 1081-4159. doi: 10.1093/deafed/env044. URL https://doi.org/10.1093/deafed/env044.

[10] J.G. Kyle, G. Pullen, F. Maddix, and B. Woll. *Sign Language: The Study of Deaf People and Their Language*. Cambridge University Press, 1993. URL https://books.google.co.uk/books?id=IrfPnQEACAAJ.

[11] Tim Ritchings PhD, Ahmed Khadragi PhD, and Magdy Saeb PhD. An intelligent computer-based system for sign language tutoring. *Assistive Technology*, 24(4):299–308, 2012. doi: 10.1080/10400435.2012.680662. URL https://doi.org/10.1080/10400435.2012.680662.

[12] Jennie E. Pyers, Anna Shusterman, Ann Senghas, Elizabeth S. Spelke, and Karen Emmorey. Evidence from an emerging sign language reveals that language supports spatial cognition. *Proceedings of the National Academy of Sciences*, 107(27):12116–12120, 2010. doi: 10.1073/pnas.0914044107. URL https://www.pnas.org/doi/abs/10.1073/pnas.0914044107.

[13] Helen Cooper, Brian Holt, and Richard Bowden. *Sign Language Recognition*, pages 539–562. Springer London, London, 2011. ISBN 978-0-85729-997-0. doi: 10.1007/978-0-85729-997-0_27. URL https://doi.org/10.1007/978-0-85729-997-0_27.

[14] Ruth Wario and Casam Nyaga. A survey of the constraints encountered in dynamic vision-based sign language hand gesture recognition. In Margherita Antona and Constantine Stephanidis, editors, *Universal Access in Human-Computer Interaction. Multimodality and Assistive Environments*, pages 373–382, Cham, 2019. Springer International Publishing. ISBN 978-3-030-23563-5. doi: 10.1007/978-3-030-23563-5_30. URL https://doi.org/10.1007/978-3-030-23563-5_30.

[15] Ming Jin Cheok, Zaid Omar, and Mohamed Hisham Jaward. A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics*, 10(1):131–153, 2019. ISSN 1868-8071. doi: 10.1007/s13042-017-0705-5. URL https://doi.org/10.1007/s13042-017-0705-5.

[16] Jieming Pan, Yuxuan Luo, Yida Li, Chen-Khong Tham, Chun-Huat Heng, and Aaron Voon-Yew Thean. A wireless multi-channel capacitive sensor system for efficient glove-based gesture recognition with ai at the edge. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(9):1624–1628, 2020. doi: 10.1109/TCSII.2020.3010318. URL https://doi.org/10.1109/TCSII.2020.3010318.

[17] Lam T. Phi, Hung D. Nguyen, T.T. Quyen Bui, and Thang T. Vu. A glove-based gesture recognition system for vietnamese sign language. In *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, pages 1555–1559, 2015. doi: 10.1109/ICCAS.2015.7364604. URL https://doi.org/10.1109/ICCAS.2015.7364604.

[18] Tim Ritchings PhD, Ahmed Khadragi PhD, and Magdy Saeb PhD. An intelligent computer-based system for sign language tutoring. *Assistive Technology*, 24(4):299–308, 2012. doi: 10.1080/10400435.2012.680662. URL https://doi.org/10.1080/10400435.2012.680662.

[19] Razieh Rastgoo, Kourosh Kiani, and Sergio Escalera. Sign language recognition: A deep survey. *Expert Systems with Applications*, 164:113794, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113794. URL https://www.sciencedirect.com/science/article/pii/S095741742030614X.

[20] Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Jonathan Michaux, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. American sign language fingerspelling recognition in the wild. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 145–152. IEEE, 2018. URL https://doi.org/10.48550/arXiv.1810.11438.

[21] Naomi K Caselli, Zed Sevcikova Sehyr, Ariel M Cohen-Goldberg, and Karen Emmorey. Asl-lex: A lexical database of american sign language. volume 49, pages 784–801. Springer, 2017. doi: 10.3758/s13428-016-0742-0.

[22] Owen Fahey, Lexset. Synthetic ASL Alphabet. https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet. [Accessed 24-Jan-2023].

[23] A.M. Martinez, R.B. Wilbur, R. Shay, and A.C. Kak. Purdue rvl-slll asl database for automatic recognition of american sign language. In *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, pages 167–172, 2002. doi: 10.1109/ICMI.2002.1166987.

[24] Vassilis Athitsos, Carol Neidle, Stan Sclaroff, Joan Nash, Alexandra Stefan, Quan Yuan, and Ashwin Thangali. The american sign language lexicon video dataset. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 0:1–8, 06 2008. doi: 10.1109/CVPRW.2008.4563181.

[25] Philippe Dreuw, Jens Forster, Thomas Deselaers, and Hermann Ney. Efficient approximations to model-based joint tracking and recognition of continuous sign language. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 1–6, Amsterdam, The Netherlands, September 2008. doi: 10.1109/AFGR.2008.4813439.

[26] Philippe Dreuw, Carol Neidle, Vassilis Athitsos, Stan Sclaroff, and Hermann Ney. Benchmark databases for video-based automatic sign language recognition. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/287_paper.pdf.

[27] Dongxu Li, Cristian Rodriguez, Xin Yu, and Hongdong Li. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1459–1469, 2020.

[28] I.A. Adeyanju, O.O. Bello, and M.A. Adegboye. Machine learning methods for sign language recognition: A critical review and analysis. *Intelligent Systems with Applications*, 12:200056, 2021. ISSN 2667-3053. doi: https://doi.org/10.1016/j.iswa.2021.200056. URL https://www.sciencedirect.com/science/article/pii/S2667305321000454.

[29] Mozhgan Kalhor, Atefeh Kajouei, Fatemeh Hamidi, and Morteza Modarresi Asem. Assessment of histogram-based medical image contrast enhancement techniques; an implementation. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0997–1003, 2019. doi: 10.1109/CCWC.2019.8666468.

[30] Ashish Sethi, S Hemanth, Kuldeep Kumar, N Bhaskara Rao, and R Krishnan. Signpro-an application suite for deaf and dumb. *IJCSET*, 2(5):1203–1206, 2012. URL http://www.ijcset.net/docs/Volumes/volume2issue5/ijcset2012020506.pdf.

[31] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987. ISSN 0734-189X. doi: https://doi.org/10.1016/S0734-189X(87)80186-X. URL https://www.sciencedirect.com/science/article/pii/S0734189X8780186X.

[32] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, 1985. ISSN 0734-189X. doi: https://doi.org/10.1016/S0734-189X(85)90153-7. URL https://www.sciencedirect.com/science/article/pii/S0734189X85901537.

[33] Muthukrishnan R. Edge detection techniques for image segmentation. *International journal of computer science and information technology*, 3:259–267, 12 2011. doi: 10.5121/ijcsit.

[34] Kankana Roy, Aparna Mohanty, and Rajiv R Sahay. Deep learning based hand detection in cluttered environment using skin segmentation. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 640–649, 2017. doi: 10.1109/ICCVW.2017.81.

[35] Partha Roy, Pradeep Kumar, and Byung-Gyu Kim. An efficient sign language recognition (slr) system using camshift tracker and hidden markov model (hmm). *SN Computer Science*, 2, 04 2021. doi: 10.1007/s42979-021-00485-z.

[36] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[37] Shagun Katoch, Varsha Singh, and Uma Shanker Tiwary. Indian sign language recognition system using surf with svm and cnn. *Array*, 14:100141, 2022. ISSN 2590-0056. doi: https://doi.org/10.1016/j.array.2022.100141. URL https://www.sciencedirect.com/science/article/pii/S2590005622000121.

[38] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. volume 3951, pages 404–417, 07 2006. ISBN 978-3-540-33832-1. doi: 10.1007/11744023_32.

[39] Madhuri Sharma, Ranjna Pal, and Ashok Kumar Sahoo. Indian sign language recognition using neural networks and knn classifiers. *ARPN Journal of Engineering and Applied Sciences*, 9(8):1255–1259, 2014.

[40] Murat Taskiran, Mehmet Killioglu, and Nihan Kahraman. A real-time system for recognition of american sign language by using deep learning. In *2018 41st international conference on telecommunications and signal processing (TSP)*, pages 1–5. IEEE, 2018. doi: 10.1109/TSP.2018.8441304.

[41] Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), 05 2016. doi: 10.1186/s40537-016-0043-6.

[42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. doi: 10.1109/CVPR.2009.5206848.

[43] Christian Vogler and Dimitris Metaxas. A framework for recognizing the simultaneous aspects of american sign language. *Computer Vision and Image Understanding*, 81(3):358–384, 2001. doi: 10.1006/cviu.2000.0895.

[44] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74048-3. doi: 10.1007/978-3-540-74048-3_4. URL https://doi.org/10.1007/978-3-540-74048-3_4.

[45] Pat Jangyodsuk, Christopher Conly, and Vassilis Athitsos. Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '14, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327466. doi: 10.1145/2674396.2674421. URL https://doi.org/10.1145/2674396.2674421.

[46] Understanding dynamic time warping. URL https://www.databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html. [Accessed 30-May-2023].

[47] Gaudenz Boesch. The 12 most popular computer vision tools in 2022, Aug 2022. URL https://viso.ai/computer-vision/the-most-popular-computer-vision-tools/. [Accessed 24-Jan-2023].

[48] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, wanghaoyang0106, Yann Defretin, Aditya Lohia, ml5ah, Ben Milanko, Benjamin Fineran, Daniel Khromov, Ding Yiwei, Doug, Durgesh, and Francisco Ingham. ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations, April 2021. URL https://doi.org/10.5281/zenodo.4679653.

[49] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. doi: 10.1109/TPAMI.2016.2577031. URL http://arxiv.org/abs/1506.01497.

[50] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[51] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *CoRR*, abs/2006.10214, 2020. URL https://arxiv.org/abs/2006.10214.

[52] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking, 2020.

[53] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

[54] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Ghum & ghuml: Generative 3d human shape and articulated pose models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6184–6193, 2020. doi: 10.1109/CVPR42600.2020.00622.

[55] Fabrizio Pedersoli, Sergio Benini, Nicola Adami, and Riccardo Leonardi. Xkin: an open source framework for hand pose and gesture recognition using kinect. *The Visual Computer: International Journal of Computer Graphics*, 10 2014. doi: 10.1007/s00371-014-0921-x.

[56] Lead Academy. What is fingerspelling, Feb 2023. URL https://lead-academy.org/blog/what-is-fingerspelling/. [Accessed 30-May-2023].

[57] Written by: Panagiotis Antoniadis. What is end-to-end deep learning?, May 2023. URL https://www.baeldung.com/cs/end-to-end-deep-learning. [Accessed 30-May-2023].

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[59] Pablo Ruiz. Understanding and visualizing resnets, Apr 2019. URL https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8. Published by Towards Data Science, [Accessed 30-May-2023].

[60] François Chollet et al. Keras. https://keras.io, 2015. [Accessed 04-Jun-2023].

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[63] William Vicars / ASL University. ASL Univeristy. URL https://asluniversity.com/. [Accessed 04-Jun-2023].

[64] Jolanta A. Lapiak. 100+ first asl words. URL https://www.handspeak.com/word/most-used/. [Accessed 05-Jun-2023].

[65] Canny edge detection. URL https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html. [Accessed 25-Jan-2023].

[66] Baby Sign Language. URL https://babysignlanguage.com/. [Accessed 09-Jun-2023].

[67] MARTIN MAGUIRE. Methods to support human-centred design. *International Journal of Human-Computer Studies*, 55(4):587–634, 2001. ISSN 1071-5819. doi: https://doi.org/10.1006/ijhc.2001.0503. URL https://www.sciencedirect.com/science/article/pii/S1071581901905038.

[68] Huzaifah A Hamid, Nur Farhana Nasri, and Norlizawati Ghazali. Colours as a form of corrective feedback in efl learners' writing. *GEMA Online Journal of Language Studies*, 18(4), 2018. doi: 10.17576/gema-2018-1804-08.

[69] Are Holen. The pbl group: self-reflections and feedback for improved learning and growth. *Medical Teacher*, 22(5):485–488, 2000. doi: 10.1080/01421590050110768. URL https://doi.org/10.1080/01421590050110768. PMID: 21271962.

[70] Jim Clark and Allan Paivio. Dual coding theory and education. *Educational Psychology Review*, 3:149–210, 09 1991. doi: 10.1007/BF01320076.

[71] Su Choe and Julian Faraway. Modeling head and hand orientation during motion using quaternions. *Soc Automot Eng*, 06 2004. doi: 10.4271/2004-01-2179.

[72] Yoichi Sato, Makiko Saito, and Hideki Koike. Real-time input of 3d pose and gestures of a userapos;s hand and its applications for hci. pages 79–86, 04 2001. ISBN 0-7695-0948-7. doi: 10.1109/VR.2001.913773.

[73] Ankur Joshi, Saket Kale, Satish Chandel, and Dinesh Pal. Likert scale: Explored and explained. *British Journal of Applied Science Technology*, 7:396–403, 01 2015. doi: 10.9734/BJAST/2015/14975.

[74] James Lewis. Psychometric evaluation of the post-study system usability questionnaire: The pssuq. volume 2, pages 1259–1263, 01 1992. doi: 10.1177/154193129203601617.

Appendix A

Appendix

A.1 Survey Questions

A.1.1 Intermediate User Survey Questions:

Key: **MC** = Multiple Choice, **S** = Slider, **H** = Hidden, **T** = Text Answer, **R** = Ranking

Qualtrics collects the following survey recipient browser information **(H)**:

- Browser Type
- Browser Version
- Operating System
- Screen Resolution
- User Agent

Q1. What level of American Sign Language (ASL) experience do you have before using SignSavvy? **(MC)**

- ○ Not knowledgeable at all (Never tried signing before)
- ○ Slightly knowledgeable (know a few basic signs)
- ○ Moderately knowledgeable (basic competency in ASL, already know most of the alphabet and some more advanced signs)
- ○ Very knowledgeable (Can communicate/understand sentences and grammar in ASL)
- ○ Extremely knowledgeable (Certified/Native/Professional ASL proficiency) (24)

Q2. On a scale of 1 (not clear at all) to 5 (extremely clear), how clear and easy to understand did you find the design and layout of the platform were? **(S)**

- *(Note: Only displayed if the response to Q2 is 3 or less)* Can you provide any additional details about specific aspects of the user interface that were unclear or confusing to you? For example, any elements or instructions that were difficult to comprehend. **(T)**

Q3. On a scale of 1 (not visually appealing at all) to 5 (extremely visually appealing), how attractive did you find the overall design and appearance of the platform? **(Slider)**

- Are there any visual aspects of the platform that you found particularly appealing or unappealing? For example, any colours, graphics, or visual elements that caught your attention positively or negatively. **(T)**

Q4. Were there any signs you struggled with in particular? If so, please list them. Also mention if you feel the sign was hard to perform because of general difficulty or because the webcam would not recognise it, even after closely following the image and text instructions. **(T)**

Q5. On a scale of 1 (crashing and lagging often, many bugs) to 5 (smooth and bug-free), how reliable would you say the app was for you? **(S)**

- *(Note: Only displayed if the response to Q5 is 3 or less)* Can you describe any technical issues you faced while using the platform? For example, any errors, glitches, or malfunctions you encountered **(T)**

Q6. On a scale of 1 (not at all useful) to 5 (very useful), how useful and engaging did you find the following when learning how to perform new signs? **(Multiple S)** *(Note: images omitted)*

- Cartoon image demonstrating signs
- Text description of how to perform a sign next to the cartoon image
- Confidence levels and current predicted sign box underneath the webcam stream
- Text guidance saying correct/incorrect sign is performed
- Image of "what you did" after incorrect sign (cropped image of hand)

Q7. Please rank the teaching mechanisms (as in Q9) in order of what you find most engaging and useful to least engaging and useful (drag and drop in order). **(R)**

- If you would like to add a comment on why you particularly liked {*top choice from Q8*}, please comment here. **(T)**
- If you would like to add a comment on why you felt that {*last choice from Q8*} was less useful or engaging, please comment here. **(T)**

Q8. On a scale of 1 (strongly disagree) to 5 (strongly agree), how strongly do you agree with the statement: "Having an interactive element in the feedback improved my ability to learn and correctly form signs compared to just watching video tutorials." **(S)**

- If you would like to add a comment about how you feel this interactive form of teaching is different to learning through videos (e.g. YouTube), please comment here. **(T)**

Q9. More generally, what additional features would you like to see from the app in the future? Do you have any suggestions on what could be added or done differently? (**T**)

A.1.2 Final User Survey Questions:

We categorise the survey results into three main categories: user interface and ease of use, system performance and reliability, and ASL demonstration and teaching mechanisms. Each category consists of five questions, rated on a scale from 1-5 using the Likert scale [73]. Participants were also given the option to provide additional details about areas that did not work well and any suggestions they might have, allowing us to gather qualitative feedback from the survey. Some questions, such as questions on ASL teaching/demonstration mechanisms or UI layouts, were supplemented with images from the application to improve clarity for participants.

Q1. Clarity of the user interface:

- On a scale of 1 (poor) to 5 (excellent), how would you rate how clear and easy to understand the design and layout of the platform were for you?
- Can you provide any additional details about specific aspects of the user interface that were unclear or confusing to you? For example, any elements or instructions that were difficult to comprehend.

Q2. Intuitiveness of tab layout:

- On a scale of 1 (poor) to 5 (excellent), how would you rate how intuitive and easy to navigate the tabbed menu for accessing different features of the platform was for you?
- Can you provide any additional details about specific aspects of the tab layout that were difficult to understand or caused confusion? For example, any challenges you faced in finding certain functions or information.

Q3. Ease of accessing learning materials:

- On a scale of 1 (poor) to 5 (excellent), how would you rate the ease with which you could access learning materials such as tutorials, videos, or documentation on the platform?
- Were there any specific challenges or difficulties you encountered when trying to access learning materials? Please provide details about any obstacles you faced.

Q4. Visual appeal of the platform:

- On a scale of 1 (poor) to 5 (excellent), how visually appealing and attractive did you find the overall design and appearance of the platform?
- Are there any visual aspects of the platform that you found particularly appealing or unappealing? For example, any colours, graphics, or visual elements that caught your attention positively or negatively.

Q5. Consistency of design elements:

- On a scale of 1 (poor) to 5 (excellent), how consistent and coherent did you find the design elements throughout the platform, such as colour schemes, fonts, and icons?
- Can you provide any examples of design elements that were inconsistent or caused confusion? For instance, any instances where different parts of the platform had conflicting visual styles.

Q6. Responsiveness of the platform:

- On a scale of 1 (poor) to 5 (excellent), how would you rate how quickly and smoothly the platform responded to your actions and interactions?
- Can you provide any additional feedback or suggestions to improve the responsiveness of the platform? For example, any delays or lag you experienced while using certain features.

Q7. Loading speed of content:

- On a scale of 1 (poor) to 5 (excellent), how would you rate the speed at which content, such as videos or images, loaded within the platform?
- Were there any specific instances where you experienced delays in loading content? If yes, please provide details and suggestions, if any.

Q8. Stability of the platform:

- On a scale of 1 (poor) to 5 (excellent), how would you rate how stable and reliable the platform was during your usage?
- Did you encounter any issues related to the stability of the platform? Please provide details and any suggestions for improvement, if applicable.

Q9. Technical issues encountered:

- On a scale of 1 (poor) to 5 (excellent), how would you rate the technical issues you encountered while using the platform?
- Can you describe any technical issues you faced while using the platform? For example, any errors, glitches, or malfunctions you encountered. Any suggestions for improvement are also welcome.

Q10. Accuracy of sign recognition:

- On a scale of 1 (poor) to 5 (excellent), how accurate did you find the system's recognition of ASL signs?
- Were there any instances where the sign recognition accuracy was not satisfactory? Please provide details about any specific signs or gestures that were not accurately recognised. Suggestions for improvement are also welcome.

Q11. Engagement and enjoyment:

- On a scale of 1 (poor) to 5 (excellent), how engaged and enjoyable did you find your overall experience with the platform?
- Were there any aspects of the platform that you found particularly engaging or enjoyable? On the other hand, were there any areas where you felt less engaged or had suggestions for improvement?

Q12. Overall learning effectiveness:

- On a scale of 1 (poor) to 5 (excellent), how effective did you find the platform in facilitating your learning of ASL?
- Did you find learning on the platform more effective than if you had watched a YouTube lesson on ASL instead? Do you have any suggestions or feedback on how to improve the learning effectiveness of the platform? For instance, any additional features or instructional approaches that you think would be beneficial.

Q13. Effectiveness of ASL demonstrations:

- On a scale of 1 (poor) to 5 (excellent), how helpful were the ASL demonstrations, such as the cartoon animations, text descriptions and videos of signs in assisting your understanding and learning?
- Are there any specific areas where you think the ASL demonstrations could be improved or any particular forms of demonstration (text, cartoon image, video) that you found more or less useful? Please provide details and suggestions, if any.

Q14. Effectiveness of teaching mechanisms:

- On a scale of 1 (poor) to 5 (excellent), how effective did you find the interactive teaching methods used on the platform for learning ASL? By teaching mechanisms, we are talking about some of the following: *(Note: Survey featured images for clarity)*
  (a) The colour-changing guidance box, giving you instructions on what to sign next
  (b) The "what you signed" image in static mode and the recorded video in dynamic mode
  (c) The predicted sign and confidence level box underneath the webcam stream
  (d) The bulleted hints on how to improve your hand and finger positions
  (e) The coloured lines on fingers that are wrongly placed and the overlay slider to see the correct sign
- Can you provide any insights on the feedback methods, such as annotations, overlays and other interactive elements, that worked well for you or any suggestions for improvement? For example, any specific ways of providing feedback or interactive tools that were particularly helpful or areas where you feel more guidance could be provided.

Q15. Comparison of teaching mechanisms:

- Please rank the following teaching mechanisms in terms of how useful and engaging you found them for learning ASL, from most to least useful:
  (a) The colour-changing guidance box, giving you instructions on what to sign next
  (b) The "what you signed" image in static mode and the recorded video in dynamic mode
  (c) The predicted sign and confidence level box underneath the webcam stream
  (d) The bulleted hints on how to improve your hand and finger positions
  (e) The coloured lines on fingers that are wrongly placed and the overlay slider to see the correct sign
- Please provide a brief explanation for your ranking decision. Why did you find the top-ranked mechanism most useful and engaging? Any suggestions for improvement?