Automatic Recognition of Simple Sentences in British Sign Language using Computer Vision and Machine Learning

CS310 Computer Science Project

Bailey Walters

Supervised by Dr. Ian Saunders

Department of Computer Science

The University of Warwick

3rd Year of Study

2022 - 2023

Abstract

Gesture recognition is a broadly researched aspect of computer vision; however, there are a limited number of works which apply it to the interpretation of British Sign Language (BSL), and in particular dynamic signs. This work explores the challenges involved in real-time sign recognition, utilising a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells to capture temporal information to aid classification. The MediaPipe library is utilised in order to create a system which is signer-independent by extracting hand landmarks from each frame of a video. We primarily explore the impact of various data pre-processing steps, and analyse common causes of confusion between signs. Although the model proposed in this work achieves only 55.7% accuracy on 6 words, the results are promising for future work.

Keywords

Sign Language, British Sign Language, Neural Networks, Recurrent Neural Networks, Classification, Gesture Recognition, MediaPipe

Acknowledgements

Special thanks go to the project supervisor, Dr. Ian Saunders, for his constant support, encouragement and advice throughout the course of the project.

Special thanks also to my partner Olviya, who, as a member of the Deaf community, provided invaluable feedback, ideas, guidance, and motivation without which this project would not have been possible.

Contents

1 Introduction

This chapter gives context to the problem, explains the problem space, and explores the motivations of this project.

In the UK, there are an estimated 150,000 Deaf people whose preferred language is British Sign Language[1]. This figure does not include the many people who may also communicate in other languages and only sometimes rely on sign language, or those people who wish to learn. Despite this, there is a lack of qualified interpreters, and many people may rely on friends, family or written text. As of 2018, there were an estimated 908 registered interpreters in the UK[2], which is less than one for every 95 BSL users.

The aim of this project is to provide a tool which is capable of recognising simple sentences in British Sign Language from a video feed in real time. Such a product is not readily available, but could have many uses. For example, communicating remotely via video calls, or giving feedback to someone who is attempting to learn. It can be very difficult to learn a signed language without practising with someone more experienced, and so if a user were able to receive feedback consistently and quickly, it could greatly benefit their learning.

1.1 Social Concerns

Although the software developed as a part of this project has no social concerns, the accompanying documentation, such as this report, must remain respectful of the people that are referred to, in particular members of the Deaf community. For example, where "Deaf" (with a capital) is used, we are referring to "people who have been deaf all their lives, or since before they started to learn to talk"[3], whereas "deaf" refers to the physical condition of having hearing loss. It is also worth noting that although BSL is primarily used by Deaf and Hard of Hearing people, it is also used by friends, family, and professionals. Throughout this report, we make clear distinction when referring to either Deaf signers or professional interpreters.

Additionally, it is important that sign language is not considered an alternative version of a spoken language. All signed languages have developed naturally, as spoken languages have, complete with their own grammar rules and regional dialects, and are therefore full languages in their own right.

1.2 Problem Space

While the focus of this project will be on British Sign Language, it is worth noting that almost every country has its own sign language. While English is the primary language of both the UK and US, British Sign Language is very different to American Sign Language. Though they contain similarities in some places, they are fundamentally different and independent, and so represent slightly different problem spaces. The most obvious difference is that ASL uses a one-handed manual alphabet, whilst BSL uses a two-handed manual alphabet, demonstrated

in Figures 1 and 2.

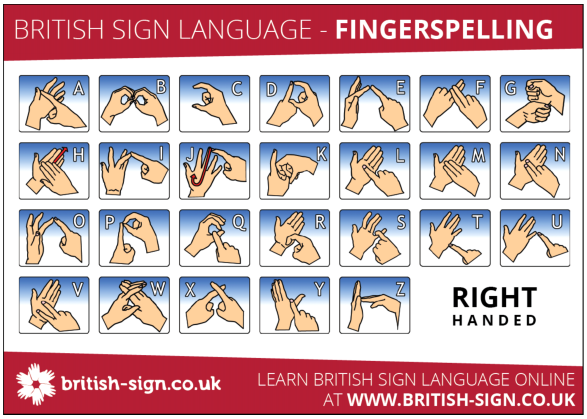

Figure 1: The BSL fingerspelling alphabet, which uses two hands.
Image from https://www.british-sign.co.uk/fingerspelling-alphabet-charts/

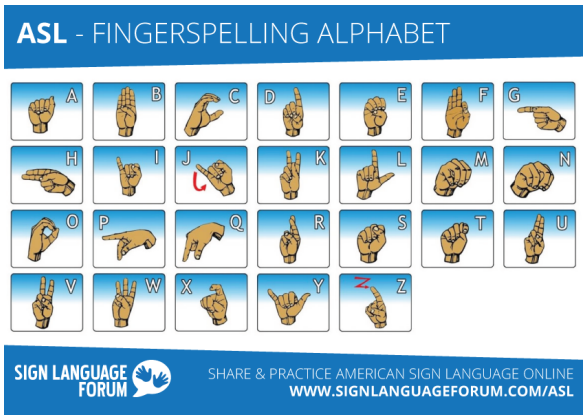

Figure 2: The ASL fingerspelling alphabet, which uses only one hand.
Image from https://www.signlanguageforum.com/asl/fingerspelling/alphabet/

In addition to there being many different sign languages, regional dialects are also present in each, just as they are in spoken languages. For this reason, BSL users from different parts of the country may not agree on a sign. There is also sometimes overlap between unique sign languages. For example, Australian and New Zealand Sign Languages are branches of British Sign Language, and so a large number of signs, including the alphabet and numerical system, are common between all three[4]. Therefore, a program capable of recognising British Sign Language could easily be extended or adapted to recognise others.

Sign languages also have many of their own grammar rules. For example, BSL does not include articles such as "a" or "the". Additionally, there are many signs which represent multiple words, such as "How are you?", which is a single sign. The structure of sentences in BSL is also different to that of English, as it is much more flexible with regards to the order of signs[5]. For these reasons, there is not a direct mapping between BSL and English, and so an automatic interpreter would require Natural Language Processing to convert a sequence of words into a full sentence.

Another essential aspect to all sign languages is facial expression and mouthing words. In addition to the context of a word, facial expressions help to convey meaning, importance, and emotion. They are also sometimes essential in recognising a sign. For example, in BSL, the sign for "please" and the sign for "thank you" are identical, differentiable only by context and the facial expression of the signer.

1.3 Project Scope

The scope of this project will be limited to British Sign Language for several reasons. The primary reason is familiarity. In addition to having some personal experience with BSL, I am close with several Deaf people who can provide feedback, guidance and advice. Secondly, there is very little research into BSL in comparison to a more widely used language such as ASL, and so there are more gaps in the literature. Finally, many signs in BSL require the use of two hands performing distinct actions, the interpretation of which provides a unique challenge which has not yet been properly addressed.

This project will aim to facilitate recognition of a small number of signs which represent different computational challenges, such as single-handed, two-handed, with hand obfuscation or similarity to other signs. The exact signs to be tested will be determined by the dataset, and discussed later in Section 4.3. This problem is fairly distinct to that tackled by other sign recognition papers, as this will go beyond simply identifying the alphabet. Whilst the BSL alphabet provides 26 distinct signs, 24 of them are static ("h" and "j"), and so can be recognised from just a single image or frame in a video. Classifying a wider variety of signs will extend this problem into time-series classification.

If time allows, this project may also include some natural language processing in order to provide a translation into sentences; however, facial analysis to aid with sign recognition is beyond its scope. This is due to both the logical complexity in interpreting expressions, and the computational complexity which would make real-time interpretation difficult.

1.4 Understanding Signs in this Paper

As the majority of online resources for sign language use videos rather than images, there are limited options available for demonstrating signs in this report. For this reason, non-signing readers are encouraged to utilise `www.signbsl.com` in order to view and understand the signs which are referred to throughout this work. Additionally, Figure 41 depicts some of the signs which are referenced throughout this paper.

2 Related Work

The problem of sign language recognition has been widely researched over a long period of time. There are many steps in the process, including image analysis, data augmentation, image classification and time-series classification. As new technologies have been developed, so have new approaches to this problem; as a result, there are many ways in which this task has been tackled. This section summarises some of these attempts, analyses the chosen approach and explains the limitations or challenges which this project aims to address.

2.1 Image Classification

The most basic signs in BSL are static. Examples include 24 of the 26 letters in the alphabet, and the numerical system. Researchers aiming to classify only the alphabet are therefore able to use images rather than videos. The first step in the process is image analysis, in order to identify the location and form of the hands.

Stephan Liwicki and Mark Everingham[6] approach BSL alphabet classification by first assigning pixels in an image a value of 'clothes', 'face' or 'background', and then utilise the colour of pixels in order to identify the hands. They then use a Histogram of Oriented Gradients to estimate the orientation and pose of each hand (see Figure 3), before Multiclass Logistic Regression is used to classify the sign. This assignment is then used with a Hidden Markov Model in order to recognise unique signs multiple frames apart in a video, and suppress classifier output of the same sign for multiple consecutive frames. The HMM is then able to output a sequence of letters which form a word. There are several distinct limitations to this work, notably that the model is signer-specific, and so must be retrained for each signer. Additionally, the dataset used is custom, and so does not represent the more varied conditions in which signing is used in the real world.

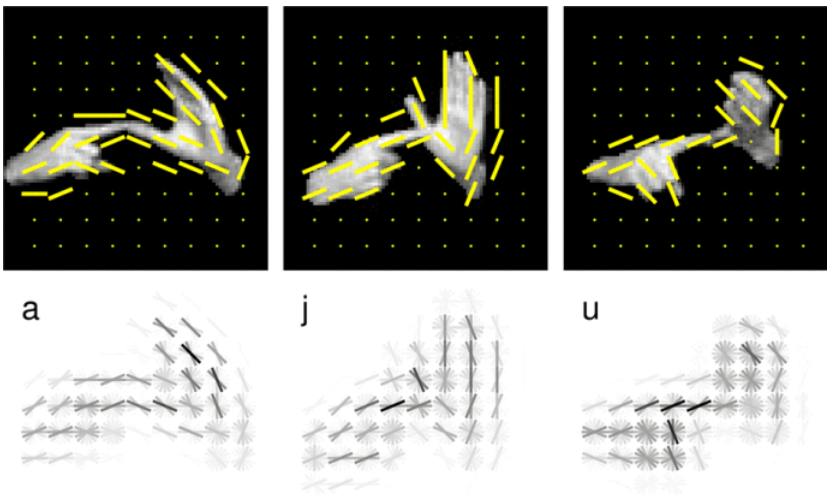

Figure 3: An example of a Histogram of Oriented Gradients applied to signs for various letters. Sourced from Liwicki and Everingham[6].

Olszewska and Quinn[7] use a very similar approach, making some assumptions to make classification easier. They assume that the given image is predominantly of a hand, and so use edge-detection techniques and a Histogram of Oriented Gradients to extract an estimated orientation and pose. A Multiclass Support Vector Machine with a Radial Bias Function Kernel is then used to classify the image into one of the 26 signs which comprise the alphabet.

Both approaches achieve an accuracy of around 99%, showing that HOG descriptors and a suitable classifier can reliably recognise static signs from a single image. Despite this, both solutions have limitations. They use custom datasets due to the low availability of BSL datasets, and so are limited to a single environment and signer. Additionally, they are only capable of recognising a very limited number of signs, and cannot be extended very far, as the majority of signs are not static.

2.2 Video Classification

The majority of research into sign language recognition systems in recent years revolves around continuous signing in order to make such systems applicable to real world situations[8]. Due to the difficulty in extracting features from images manually, deep learning is the preferred method of classification. Some deep learning techniques that can be applied to the problem are Convolutional Neural Networks, Generative Adversarial Networks and Recurrent Neural Networks.

One such example is found in *Video-Based Sign Language Translation System Using Machine Learning* by Sonare et al.[9], which uses both a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) for automatic recognition of signs in ASL. The proposed approach achieves an accuracy of 90.1% over 10 signs. The CNN is first used for feature extraction from depth images, before the RNN is used for extracting temporal patterns and finally a Multi-Layer Perceptron is used for classification.

K. Kumar proposed another approach in *DEAF-BSL: Deep Learning Framework for British Sign Language Recognition*[10], in which a 3 dimensional convolutional neural network is used for sign classification over multiple frames of video. Several pre-processing steps are performed on each frame, such as colour augmentation, median filtering, Gabor filtering, Sobel edge filtering, and Otsu thresholding. Despite this, the model is trained on only the BSL alphabet, which, as previously stated, is largely static. For this reason, there is little evidence that the proposed solution will scale to dynamic signs, although it does appear to be superior to a Hidden Markov Model for recognising sequences of signs, with close to 99% recall for all letters.

Elakkiya, Vijayakumar and N. Kumar[11] propose a solution which uses both a recurrent neural network and a 3D-CNN for sign language classification. They use these models to form a Hyperparameter based optimised Generative Adversarial Network, in which the output is the sign classification. The RNN using LSTM nodes is a generator, and the 3D-CNN combined with another LSTM layer is the discriminator. From the paper, “The generator generates

random sequences with noise from the real sequence of frames, and the discriminator detects and classifies the real frames of sign gestures." This leads to a highly complex solution which is capable of 100% accuracy with signer dependence, or 98% with signer variance.

One of the most extensive works surrounding British Sign Language recognition is *Automatic Learning of British Sign Language from Signed TV Broadcasts* by Beuhler[12]. As the title suggests, this work additionally suggests an unsupervised learning system using TV broadcasts of signers with subtitles, which is beyond the scope of this project. The paper details many pre-processing steps, such as hand, arm, head and torso identification. This involves a variety of techniques, including the previous described Histogram of Oriented Gradients. Finally, a Multiclass Support Vector Machine using only the first and last frame of a given sign is used for classification. Note that the dataset used is called BBC-Oxford British Sign Language Dataset, or BOBSL, and contains footage of many signers with a variety of backgrounds (see Section 4.3.2). The classifier proposed, which uses features representing hand trajectory, hand shape and hand orientation, can achieve an accuracy of 57.9% on 91 dynamic signs in a signer-dependent setting, or 39.5% in a signer-independent setting. Compared to random guessing, which would have an accuracy of 1.1% for the 91 signs, this is a significant milestone.

While this research shows that it is possible to train a model on a large number of dynamic signs with a high degree of accuracy, it is evident that it has many shortcomings. Notably, the fall in accuracy when multiple signers are introduced indicates that skin colour is likely a large factor in the model's ability to recognise signs, as the professional signers in the dataset have very similar signing patterns. Additionally, due to the significant amount of image processing, and 91 1-vs-rest SVM classifiers involved, it is unlikely that classification is in real-time. This project hopes to improve upon this work by developing a system which is signer-invariant, making use of tools which have become available since this paper was written.

2.2.1 Works using MediaPipe

In recent years, Google have developed an open-source all-in-one style machine learning library called MediaPipe[13], which provides machine learning models for solving many computer vision applications, such as object detection, pose landmark detection, face landmark detection and hand landmark detection. As this technology is cutting-edge, there are few papers utilising it for sign language interpretation. Two that do are *Sign Recognition Using Deep Learning*[14] and *MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition*[15]. Each of these use MediaPipe's landmark detection in combination with a neural network to recognise signs in a video sequence.

Ray et al.[14] trained three individual ANNs for recognition of the ASL alphabet, the first to recognise any letter, and each of the others specifically to combat ambiguity in signs. The three networks are then used to classify every frame of a video sequence, with a sign only being accepted if it is present in 20 consecutive frames. This is a rudimentary approach to continuous

sign recognition, but does showcase the power of MediaPipe, achieving 94% accuracy over the 26 ASL alphabet signs.

Samaan et al.[15] also utilise MediaPipe; however, they use a Recurrent Neural Network for sign classification, as it can use temporal patterns to classify dynamic signs. They claim an accuracy of 100% over 10 dynamic signs using Gated Recurrent Units, and 99% with Long Short-Term Memory (LSTM) and BiLSTM units. Both Mediapipe's hand and pose recognition are used in order to gather the data for classification. They additionally test with facial landmarks, achieving the same accuracy score, but with a significantly slower model due to the computational complexity. The dataset used for this research contains consistent videos, with various signers in a room with good lighting conditions and a stationary background. Additionally, video clips in the dataset are all the same length and contain the same number of frames. While these conditions are ideal, they are not representative of real situations. It is also unclear which sign language is used, or how dynamic the signs involved actually are.

This project hopes to build upon these studies by creating a system which is capable of recognising dynamic signs in more varied conditions. We also aim to explore which characteristics of signs cause a loss in accuracy, such as faster movement, smaller movements, or the use of two hands.

3 Objectives

With an understanding of the problem space, and knowledge of existing works, we are able to define a set of objectives which this project should meet. These objectives will break down the overall task, which is to create a tool capable of translating signs to text in real time. Each objective falls into one of three categories: must-have, should-have, or could-have. Throughout this report, objectives will be referred to and compared against for progress, before the success of the project is analysed in the Conclusion using these objectives. The objectives below are as stated in the Project Specification[16].

The objectives which are considered must-haves are:

M1. The program must be able to access both live video feed and a recorded video.

M2. The program must be able to accurately locate the hands of people in an image, even on low-quality cameras such as a webcam.

M3. The program must use machine learning to attempt to translate any signs to text.

M4. The program must be able to translate in real time as a signer on screen is signing.

These objectives aim to define a program capable of filling the gap which is currently present in research - the accurate translation of dynamic signs in real time.

The objectives which are considered should-haves are listed below. They aim to ensure that the result of this project is accurate and easy to use.

S1. The program should be able to identify a small selection of signs (~ 20) with an accuracy greater than 90%.

S2. The program should be able to translate signs in varied conditions, including lighting, distance of the signer from the camera, and camera quality.

S3. The program should be able to use multiple frames from a video to help classify a sign.

S4. The user should be able to change the camera source with ease.

S5. The program should be able to identify the hands of people with varying skin colours.

S6. The program should form sentences out of the signs that have been recognised.

These objectives also serve to differentiate this project further from existing work. Where much previous research focuses on the use of individual frames to classify a sign, it is important that this project can make use of all the data available from a video feed. Similarly, a signer-independent solution will stand-out among similar works.

The following objectives are not vital to the success of the project. However, they would help with accessibility and usability. These objectives are considered could-haves:

C1. The program may be able to identify signs by one signer where there are multiple people on the screen at once.

C2. The program may be expanded to recognise a larger number of signs.

Together, these objectives define the structure, and goal of the project. They are used to justify design and methodology decisions and are important in determining the success of the project. As not all are considered necessary, some will be prioritised above others, and as such will be the focus of sections of this report.

As the aim of this project is to create an interpretation tool, the majority of the defined objectives can be considered functional. This means that they define things that the system should do, with less of a focus on overall properties of the system. If this project were expanded to create a complete software application, other objectives revolving around user experience should be included. However, this is not part of the current scope.

4 System Design and Methodology

This section will explain in detail the methodology and design decisions for this project. Each design decision can be justified and explained with reference to the problem space, the background research, and the objectives. The design of the system is crucial in developing software capable of meeting the objectives, and is influenced heavily by related works. The design chosen uses knowledge gathered by others in order to overcome the difficulties they have faced, and in doing so, meet all objectives. It also minimises the risk of implementation issues by creating a defined and logical structure which can be referred to throughout development. The design of the system in turn influences the development methodology used, as this is an important factor in ensuring that tasks are effectively prioritised and completed in a timely manner.

4.1 Development Methodology

The development methodology chosen for this project is an agile, iterative approach. There are several reasons for this, primarily the fact that this enables consistent and repeated learning opportunities. As the technologies and approaches used are relatively new, it is important that prototypes are made which can influence further development. This will take the form of repeated adjustments to the data, neural network, and other technologies. The execution of this methodology largely uses Basil and Turner's description[17], where the objectives resemble a *project control list* (a list of individual components which comprise the final product). Each iterative step consists of selecting a task, designing it, implementing it, and then analysing the changes which were made. We adapt this methodology to better fit the project by focusing each iteration on the improvement of one aspect of the system, rather than implementing a new component.

The three primary factors which each iteration will address are:

1. The number of words/signs for the model to be trained on.
2. The sample size for each word/sign.
3. The design of the neural network.

Initial development will focus on creating a minimum viable product, with a small selection of words and a simple machine learning model. The sample size of data will be small, in order to ensure that development time is minimised, and familiarity can be gained with the different technologies involved. The initial prototype must also implement the foundational system for detecting hands in a video and represent this data in a usable format. After this, each iteration will further develop a minimum of one of these features in order to build up the system gradually and ensure constant improvements. Testing will be more thorough for later iterations of the project in order to enable more fine-tuned analysis of the system, as these will be closer to the final result. Early iterations will focus on significant improvements.

An iterative approach also allows for frequent analysis of the technologies used and surrounding research. The MediaPipe library (see Section 4.4.2) is frequently updated, and several papers utilising it have been released during the course of this project[14, 15]. It is important that we are able to utilise the most recent technologies and knowledge, and so continued background reading is performed throughout development.

This methodology also minimises the impact of time constraints. The duration of each development cycle can vary, meaning that if the project falls behind schedule, more improvements and changes can be made in a single iteration to compensate. This also means that there will always be a functional version of the project available for demonstration or testing.

4.1.1 Time Management

Due to the uncertainty in the number of iterations that may be required, it is difficult to plan a complete schedule. The project was therefore broken down into the core components, with significant development time being allocated to iterative improvements in the system. The initial timeline is represented by Figure 42, which is as seen in the Project Specification[16]. Throughout the course of the project, this Gantt chart is referred to as a measure of progress, alongside the objectives. If at any point the project is behind schedule, the timeline may be revised and adjusted.

In order to ensure that the overall timeline adheres to the schedule, it is also important that the day-to-day schedule is well defined. For this reason, development or research should be continuous, with a minimum of several hours every week. This will allow not only continued development, but consistent reflection of progress over time. Time for the project will be allocated depending on the weekly course structure and time available, although more flexibility will be introduced during the holidays and around other coursework deadlines.

The effectiveness and execution of this time management approach will be analysed in Section 7.2.

4.1.2 Management Tools

Throughout the project, several tools were chosen to manage the process of software development. These tools will allow for swift and painless iterative development, with consistent backups and code availability.

GitHub: GitHub is a free online tool for hosting code repositories[18] with millions of users. In this project, it will be used for version control, remote backups, and code availability. As development may occur on multiple systems, GitHub will allow for file consistency no matter the device. The branching system will also allow for simple implementation and testing of new features without jeopardising the integrity of the last known working version.

Overleaf: Overleaf will allow for online development of documentation using LaTeX. This ensures that the format of all documentation is consistent, and that it can be updated from anywhere.

OneDrive: Microsoft OneDrive is used as a secondary backup system. OneDrive will sync files automatically to the cloud, whereas GitHub requires them to be manually pushed. OneDrive will therefore ensure that the most recent version of the code is always available on any device, even if it has not yet been pushed to GitHub.

4.2 System Architecture

The overall architecture of the system is informed by knowledge of other systems, as well as the objectives which have been laid out. The project can be broken down into a components diagram as displayed in Figure 4, which is as seen in the Progress Report[19], with some corrections to the flow.

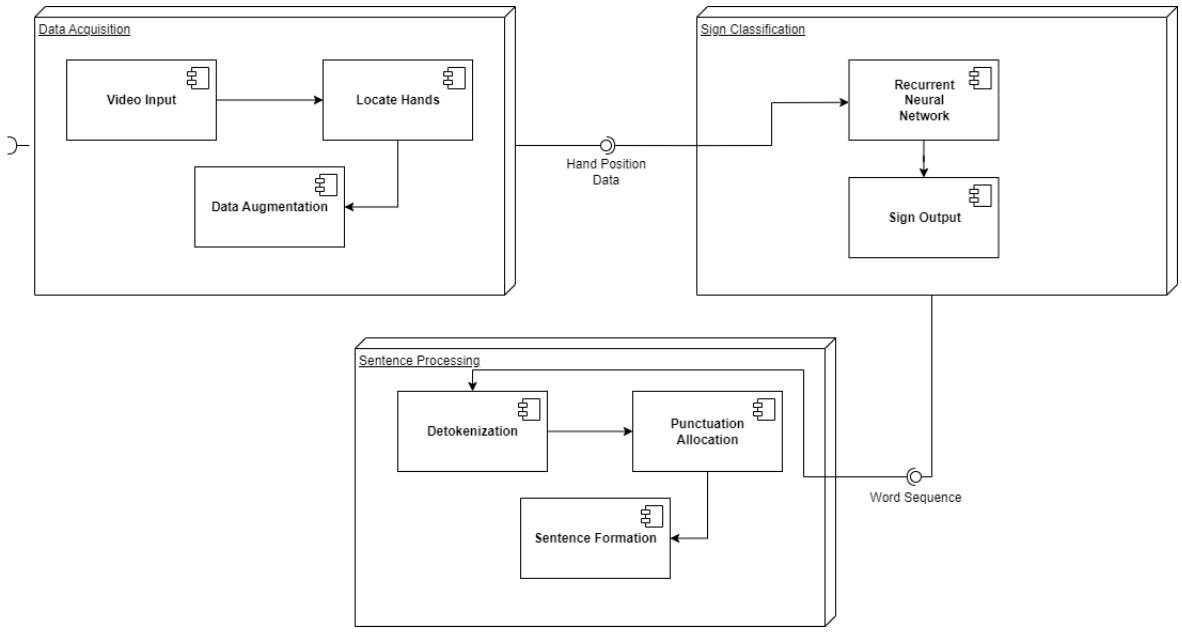

Figure 4: Components Diagram

The components diagram represents not only the components of the system, but the flow of data through it. Each component is responsible for a significant part of the system, in which data is received, processed, and output to either the next component or the user. The function of each component is described below:

Data Acquisition: The Data Acquisition component of the system is important in ensuring that the data available to the machine learning model represents the key features which can be used to classify signs. This component is responsible for receiving a video, locating the hands in each frame, and augmenting the data as necessary. As this component uses MediaPipe[13] to locate and track hands in the video, little data augmentation is necessary

(see Section 4.4.2). This component's primary function will therefore be ensuring that the correct number of frames are recorded, any missing data is estimated, or the sample is discarded if hands are not found in the video. The data will then be passed on to the Sign Classification component, ready for a new video clip to be processed by the Data Acquisition component.

Sign Classification: The Sign Classification component is responsible for using a sequence of coordinates representing hand landmarks to classify the sign. The data it receives may be of variable length, as signs may take varying amounts of time to perform, and so a Recurrent Neural Network will be used (see Section 4.5.2). Each array of data representing hands in a frame which is passed to the network will be used to help classify the sign using all previous information it has received. When the network predicts a sign with confidence above some threshold, the sign is output to the next component in the system. The confidence threshold will be determined through testing.

Sentence Processing: Over the course of a video sequence, the Sign Classification component should output a series of signs. The function of the Sentence Processing component will be to apply English grammar rules to the list of words in order to form a complete sentence. For the purposes of this project, sentences will be limited to a single clause, with only a full stop as punctuation. This component was unfortunately not implemented due to time constraints (see Section 7.2).

This system architecture will be used for both training and application. In the training stage, the video input will be short, pre-recorded clips with a label which will be used to train the neural network. Once trained, the video input will be either a live video feed or a pre-recorded video, which will be passed through the Data Acquisition component in order to retrieve hand position data, which can then be used to predict a sign. In a live video feed, a fixed number of frames will be passed to the neural network, for example the 30 most recent frames.

4.3 Data

The data on which a machine learning model is trained plays a huge factor in its performance. There are several factors governing the quality of data, which must all be considered when determining which data source to use. These factors, in relation to this project, are described below:

- **Label Correctness:** It is imperative that the data which is used to train a machine learning model is labelled correctly. In the best case, there is no mislabelled data, although it is likely that there are some mistakes, in which case the model should be able to learn that they are outliers to be ignored. If there are a significant number of incorrect labels,

a model may either learn to categorise something with a completely incorrect label, or effectively pick at random if there is no clear label for all instances of similar data. As many signs in BSL are fairly similar, it is important that all of the data is correctly labelled. For example, the signs for letters "a" and "e" are very similar, with the index finger of the dominant hand pointing to a different finger on the non-dominant hand (see Figure 1). If the training data does not make a clear distinction between these two signs, it is likely that the model would struggle to differentiate.

- **Sample Size and Variation:** The number of samples for each class is also very important. As there will be slight variations in the data, a larger number of unique samples allows the machine learning model to learn the features which connect them, and thus the features which represent the class. This is particularly pertinent to the problem of sign language where the signer varies. Just as everyone has their own mannerisms, every signer will perform signs slightly differently, in addition to having slightly different height, arm length, hand size, etc. A neural network must be able to learn that these features are not relevant to the classification of the sign, which is easier if there are more samples for a given class which are each different in some way.

- **Video Quality:** As the format of the data required for this project is video, it is important that the training data is as consistent and high quality as possible. For example, a higher resolution video will give the MediaPipe hand detection model a better chance of finding each hand landmark in a frame, while a video with a higher number of frames per second will help to reduce motion blur, which will have the same effect. More frames per second also means that more data is available to train the model, although it is important that the system is able to adjust the frame rate of a video such that the frame rate is standardised. The system could also be adapted such that a video is sped up or slowed down to match the speed of signing to that which the model was trained on.

As previously discussed, there are approximately 87,000 Deaf people who use BSL on a daily basis[1]. In comparison, there are an estimated 500,000 Deaf users of American Sign Language[20]. This has led to significantly more research into ASL, and very few publicly available datasets for BSL. The only two video datasets publicly available are the BSL Corpus Dataset[21] and the BBC-Oxford British Sign Language Dataset[22]. The majority of researchers develop their own datasets; however, this has its own challenges. Each of the options for data are discussed below.

4.3.1 BSL Corpus

The BSL Corpus Dataset is a dataset compiled by University College London[21]. It is comprised of over 1000 videos of 249 Deaf users of BSL. These videos are broken down into conversations, narratives, interviews, and lexical elicitation. Additionally, the signers come from

across the UK, resulting in different dialects being present in the dataset, and have a wide range of ages. This is the dataset used by Kumar[10] for recognition of the alphabet in video sequences.

Although this dataset contains a huge number of samples, there are several factors which make it unsuitable for this project. For example, the vast majority of videos involve two people sat at an angle to the camera. While it would not be a problem to crop the video feed to isolate one signer, the model would learn the motion at a specific angle, which would be difficult to reproduce, and make the model less useful for real world situations. It may be possible to perform rotations in 3D space, such that the model can train on data which is at various angles. The depth values generated by MediaPipe are only estimates and are not relative to the frame of the camera. This is discussed further in Section 4.4.2. It may also be possible to use multiple datasets in order to train the model to recognise signs at different angles. Additionally, the presence of regional dialects introduces some complexity in terms of label correctness. This could be avoided by limiting the samples for words with regional variations to a single version of the sign.

The largest issue with the dataset is the lack of structure, subtitles, and mappings between signs and words. Very few of the videos in the dataset have accompanying subtitles or translations, and there is no method of automatically finding videos containing a specific word. Therefore, the data would have to not only be labelled manually, but also found and extracted manually. Therefore, this dataset is unsuitable for this project.

4.3.2 BBC-Oxford British Sign Language Dataset

The BOBSL dataset is a collection of videos from televised BBC broadcasts[22]. There are 1,940 individual episodes, constituting approximately 1,400 hours of signing footage. This dataset contains a vast collection of words, all of which are present in subtitles, allowing every sign to be labelled. There are 37 distinct signers in the dataset, allowing development of a signer-independent system. As the data is from professional BBC signers, the signs are consistent without regional variations, and every signer is a fixed distance away from the camera. This means that variation in movements will represent differences in the signer, rather than conditions such as camera distance, allowing a machine learning model to more accurately train on the movement of the sign.

This dataset also has some problems. Due to the age of the data, and the fact that it is cropped from the original broadcast, the camera quality is relatively low, and contains significant motion blur. This is a fact that was identified by Buehler[12], and limited the ability of his model. Using MediaPipe, the problem of low resolution is less significant, as the hand detection model is trained to recognise hands even in difficult images. It is not able to identify hands where there is motion blur, although this can be worked around by estimating hand positions or using the last known locations.

Another potential problem is the fact that the background behind the signers is moving due to the contents of the broadcast. Traditionally, this can cause issues in differentiating the signer from the background, particularly when the signer's clothing is a similar colour; however, the MediaPipe model is able use the last known location of landmarks to estimate where their next position will be, minimising the impact of moving backgrounds.

The deciding factor in whether to use this dataset or not was the accompanying documentation. The dataset contains a file called *spottings.json*, which contains over 18 million lines detailing in which video every word appears, where it appears, and the probability that the classification of that word is correct (see Figure 6). There is also an interface for automatically downloading videos via a script, which enables automatic downloading and selection of videos as required in order to maximise storage space efficiency. For this reason, this is the dataset chosen for this project initially. Unfortunately, it would later become evident that the quality of the data was not as expected, and in fact many samples were mislabelled completely. This is discussed further in Section 6.

Figure 5: A sample video frame from https://www.robots.ox.ac.uk/~vgg/data/bobsl/.

```
{
    "train": {
        <word>: {
            "probs": [...],
            "global_times": [...],
            "names": [...]
        }
    }
}
```

Figure 6: The structure of the *spottings.json* file for the BOBSL dataset.

4.3.3 Custom

As previously discussed, many researchers choose to make their own datasets. This has several benefits, as you are able to format the data in such a way as to make development easier, and have complete control over the quality and sample size. It does also have some limitations which could make it unsuitable for this project. Firstly, the speed and competence of signing is dependent on the person signing, and so a custom dataset in this instance would likely not be entirely representative of someone who has been signing for an extended period of

time. Additionally, it may be difficult to control the variations in samples in such a way as to generate a good distribution of people signing in different conditions, while keeping some variables constant like camera quality and the frame rate of the video. Another issue with this approach is that it is very time consuming. In order to maintain consistent quality of the data, each sign must be performed carefully many times.

For these reasons, a custom dataset will not be used for this project unless other datasets prove to be sub-par or contain significant problems.

4.3.4 Other Languages

Another option is to use a dataset from another language, such as ASL. Although this would be different to BSL, there are many similarities in the computational problem being solved. Both languages involve dynamic movement across multiple frames of a video to create a sign, and so a model capable of recognising ASL could be extended to BSL. The primary difference is that many ASL signs require the use of only one hand, as opposed to two in BSL, however there are many signs in BSL that also use only one hand, and so it would not be entirely unrepresentative.

Some publicly available datasets that could be used for this project are ASLLRP[23], ASLLVD[24] or WLASL[25]. These will only be used if the BSL datasets are not sufficient to create a functional model, and all other options have been exhausted, due to the distinct differences in the languages.

4.3.5 Legal Concerns

Due to the use of the BOBSL dataset[22], there is one legal concern that is considered throughout the project, which is the storage and use of the data. As per the agreement which allowed use of the dataset for this project, it must be stored securely on the University of Warwick's servers, and cannot be shared or distributed. The login details also cannot be shared or distributed. Therefore, the accompanying code submission will contain neither the data, nor the login information required to access it. Furthermore, the machine learning models developed in this work cannot be used for commercial purposes.

4.4 Hand Detection

In order for a machine learning model to classify signs, it is important that the system is able to extract an accurate representation of a person's hands from a video. This means that each frame of a video needs to be analysed, and the position of both hands tracked throughout.

There are two possible approaches to solving this problem. The first is to use image augmentation and analysis techniques in order classify each pixel of an image as either 'hand' or 'non-hand', and then use knowledge of whether a hand is left or right to create a representation which is useful to a machine learning model. The second approach is to bypass the majority of

the image processing step by using an existing library such as MediaPipe[13] to locate hands in an image.

4.4.1 Image Analysis

In order to use image analysis to extract hands, the system must first understand the colour of the signer's hand. In order to do this, existing works first use face detection techniques in order to create an estimated model of the skin colour. Liwicki[6] additionally creates models for clothing colour and background colour, and applies a spacial prior to detect the hands. Buehler[12] uses similar colour model techniques, manually creating colour models for the skin, instead of generating them automatically, and then using body pose estimation techniques in order to localise the hands, wrists and arms.

Both of these approaches then apply Histograms of Oriented Gradients[26] in order to detect edges. A Histogram of Oriented Gradients uses local intensity gradients and edge directions in order to detect not only external, but also internal edges. In this way, a representation of the orientation of the hand is created, as well as the orientation and shape of the fingers, which may create internal edges to the shape of the hand. Figure 7 demonstrates the process used by Buehler to identify hand pixels using a colour model, before applying a Histogram of Oriented Gradients in order to find a numerical representation of the hand shape.

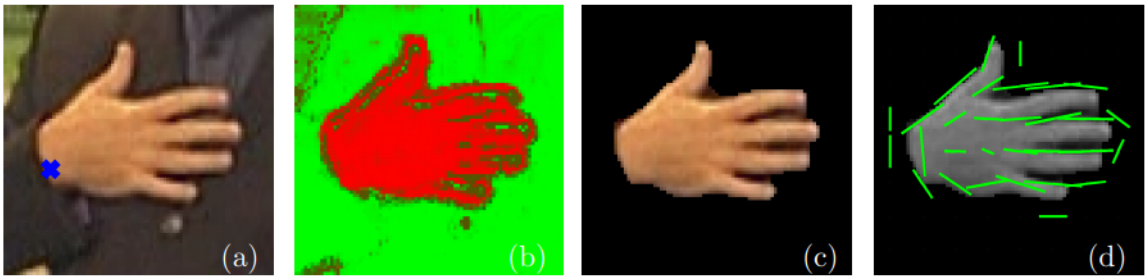

Figure 7: An example of a HOG (d) applied to an image of a hand after colour segmentation (b).
Image from *Automatic Learning of British Sign Language from Signed TV Broadcasts*[12]

The upper-body model used by Buehler proved very effective for localising hand, wrist and arm positions, however it requires prior knowledge of future data. Distinctive frames are used where the upper body pose is clear, before estimation and sampling techniques are used to infer the position of the arm in the frames between.

4.4.2 MediaPipe

Instead of manually applying colour models and other image analysis techniques in order to locate the hands in an image, it is now possible to use the MediaPipe library[13]. This library is a collection of machine learning models and tools capable of performing complex tasks. The

library is available for Android, Python, or JavaScript, which means it is widely available and could still be used if the project is later moved to or further developed for another platform. Note that this project uses MediaPipe version 0.8.11, which is now considered legacy. While the API described may change, the functionality remains the same.

MediaPipe contains four machine learning models which could be useful for this project. These are hand landmark detection, pose landmark detection, face landmark detection, and holistic landmark detection, which is the previous three combined. Each of these models extracts some number of landmarks from an image, such that the landmarks form the shape of the target feature. The landmarks returned from the model are given as coordinates of the form (x, y, z), where x and y are normalised to the range $[0.0, 1.0]$, and z represents the relative distance from landmark to some other central landmark, with respect to the camera.

The hand detection model will be most pertinent to this project, as many signs are dictated primarily by movement of the hand and fingers. Although many signs also use other parts of the body, such as the arms or the head, the movement of the hands relative to one another will often inherently represent the use of other body parts. Therefore, the majority of this project will focus on use of hand landmarks, although this may be expanded to include pose landmarks as explored by Samaan et al.[15]. The model is extremely powerful, capable of finding hands in an image, video, or live stream. It is comprised of two sub-models, the first being palm detection, and the second being landmark detection. Palm detection is used to localise the hands in the image, before landmark detection is performed on the relevant area. When applied to video or live streams, the model is able to utilise the persistence of hands throughout frames to estimate the locality of each hand. In this way, the palm detection model is only used if the landmark model is unable to find hands in the expected space, reducing computation time.

The hand landmark detection model outputs 21 unique landmarks for each hand, many of which are placed on joints, as shown in Figure 8. This means that the angle between landmarks represents the angle of the finger at that joint, thus capturing the shape of the hand. The z coordinate representing depth is relative to the wrist for this model, which is landmark 0. Smaller values indicate that a landmark is closer to the camera, and larger values indicate it is further away.

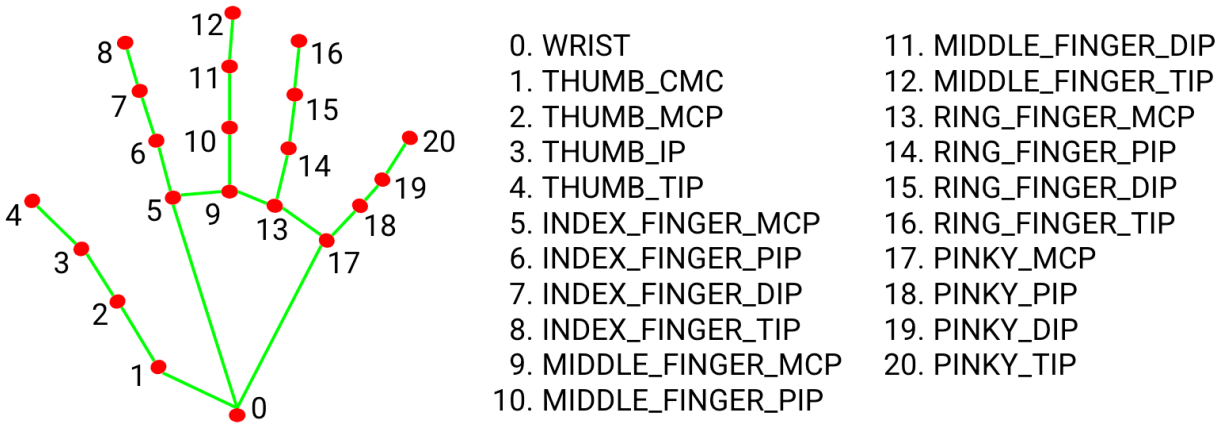

Figure 8: MediaPipe hand landmarks. Image from MediaPipe documentation[13]

The format of this data accurately represents not only the shape of the hand, but also its orientation, and position in the frame. In this way, it provides equivalent representative power to using a HOG descriptor on an image of a hand, whilst requiring fewer manual image processing steps.

The pose landmark detection model uses a very similar format to the hand landmark detector, finding the most prominent person in the first frame, and then tracking them until their pose is no longer detected. The output is also very similar, with the depth value relative to the estimated centre of the hips. Each landmark is also given a visibility value, which is the likelihood of it being visible in the frame. The landmarks are shown in Figure 9.

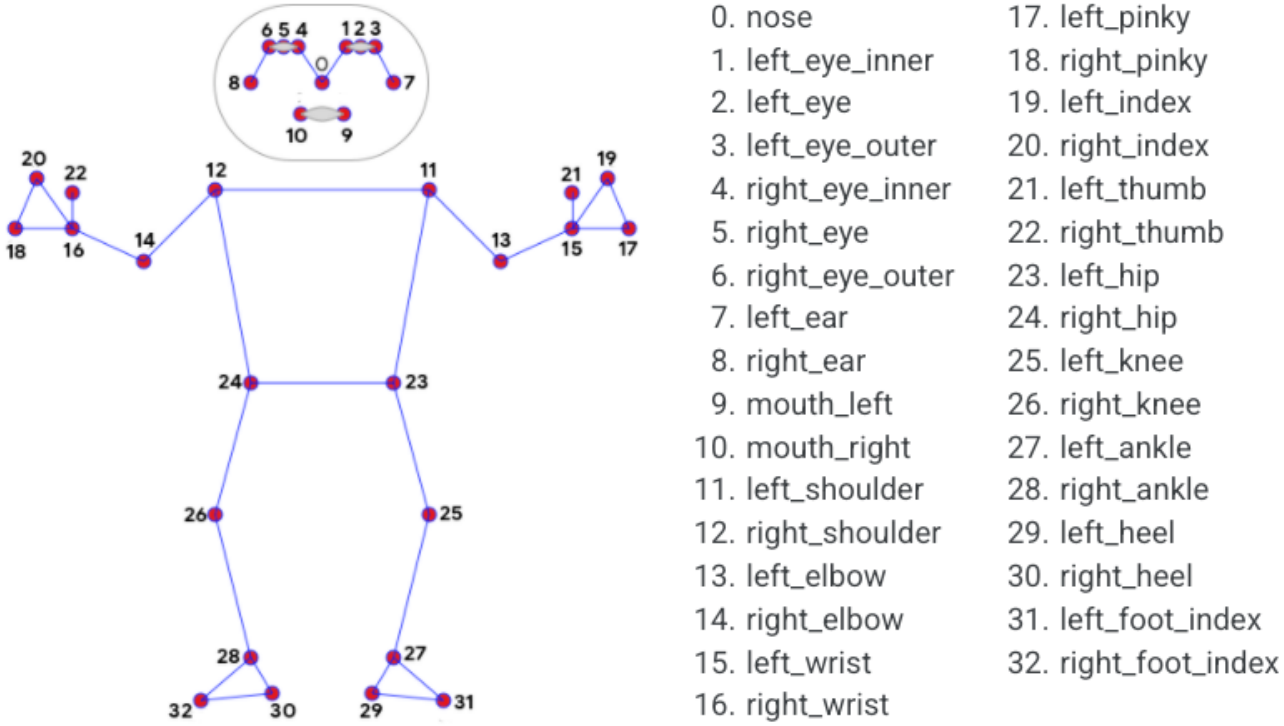

Figure 9: MediaPipe pose landmarks. Image from MediaPipe documentation[13]

Pose information could help the model to learn the importance of hand movements relative to the body, rather than relative to the frame of the picture. This may not be necessary for the data used, as the signers are a consistent distance from the camera, and so the frame is already relative to their body. If time allows, testing will be performed to see if including pose data improves the quality of the model developed.

The face landmark detection is significantly more powerful and complex than the hand or pose detection, as it can detect 468 different landmarks, and generates 52 blendshape scores representing facial expression. Unfortunately, this high complexity makes the model difficult to use in real-time systems, although it would likely be beneficial for the classification of some signs which require mouthing, as discussed in Section 1.2.

The MediaPipe python library will be utilised for automatic hand detection in this project, using the coordinates returned by the model as features of the data. Each frame will therefore have a minimum of $42 \times 3 = 126$ features, as each coordinate has three components. If pose detection is added, this count increases to $126 + 33 \times 4 = 258$. If a sign takes one second, at a framerate of 24 frames per second, then there will be at least $24 \times 126 = 3024$ data points to be used for classification.

4.5 Sign Classification

This section will detail the technology decisions and network design for the machine learning model. This will take into account works discussed in Section 2, as well as the structure of the data and desired result.

4.5.1 Potential Models

As seen in the background reading, many different models have been created for the classification of signs, yet very few of them are applicable to this project. Many approaches classify every frame of video individually, and use external logic to determine the sign which is present throughout the frames[14, 6]. Even Buehler[12] uses only the first and last frame of a sign to classify it. This technique, while effective for signs with little movement, cannot effectively capture signs with dynamic movement of the arms or hands. The models used in these papers are a feed-forward ANN, Multiclass Logistic Regression, and a Multiclass Support Vector Machine. All of these models specialise in linear data, which is not applicable to dynamic signs, which may be of variable length.

Kumar proposes an approach that makes use of a 3-Dimensional Convolutional Neural Network[10]. 3D-CNNs are based on work by Tran et al.[27], and are able to use multiple frames of a video for classification tasks. They are functionally similar to a 2D CNN, but allow for temporal influence as well. The approach taken by Kumar uses 5 of every 25 frames for classification, using images that have been passed through many filters and edge detection algorithms. In this way, the model is able to achieve a high accuracy, however it is only applied

to the BSL alphabet, which is largely static, and so it is unclear as to whether the model benefits from the temporal data provided by multiple frames.

Another approach that several researchers have made use of is Recurrent Neural Networks[9, 15]. In an RNN, connections between nodes can create cycles, meaning that the output from a node can affect subsequent inputs to the same node. The presence of these cycles also means that RNNs can be used for variable length inputs, making them very popular in language related tasks. Sonare[9] uses a CNN for feature extraction from each frame after image pre-processing, the output of which is passed to an RNN constructed from Long Short-Term Memory units. The RNN is then able to use information from multiple frames in order to extract temporal patterns and classify the sign. A caveat to this work is that the model was used only for classification of the alphabet, and so temporal information was likely limited due to the lack of movement in the signs.

Samaan et al.[15] proposed a similar approach, also utilising the power of Recurrent Neural Networks. They use MediaPipe for extracting hand and pose information from each frame, and pass the model exactly 30 frames of data each time. They also train and test three different models, using Gated Recurrent Units (GRU), as well as LSTM and Bi-LSTM, which is a bi-directional version of an LSTM node. All three proposed models achieve very high accuracy, indicating that they are all able to utilise temporal data. The accuracy achieved, and the classification of "dynamic signs", indicate that a Recurrent Neural Network is a good choice for this project, and would allow verification of the claims made about its accuracy.

After considering all possible approaches, it was decided that a Recurrent Neural Network will be used for this project. It allows temporal information from hand movement to be used in classification and has the additional benefit of allowing variable length inputs, which can be useful when the length of a sign is unknown.

4.5.2 Recurrent Neural Networks

Recurrent Neural Networks are a form of artificial neural network in which connections can form cycles. This means that the output from a node may be used to affect subsequent inputs to that same node. The nature of this recurrence means that the cycle can occur any number of times. For this reason, RNNs are commonly used for classification and time series prediction[28] on variable length data.

The most basic form a Recurrent Neural Network is one in which each neuron sums the inputs together with the output from the neuron at the previous time step, and then applies an activation function. Figure 10 demonstrates a simple Elman RNN[29] which employs a ReLU activation function in each neuron. An Elman network is one in which the output of the neuron is used directly in the next time step, as opposed to the output from the model at that previous time step, which would be a Jordan network[30]. This passing of information along time steps allows the model to use information from all previous time steps for classification.

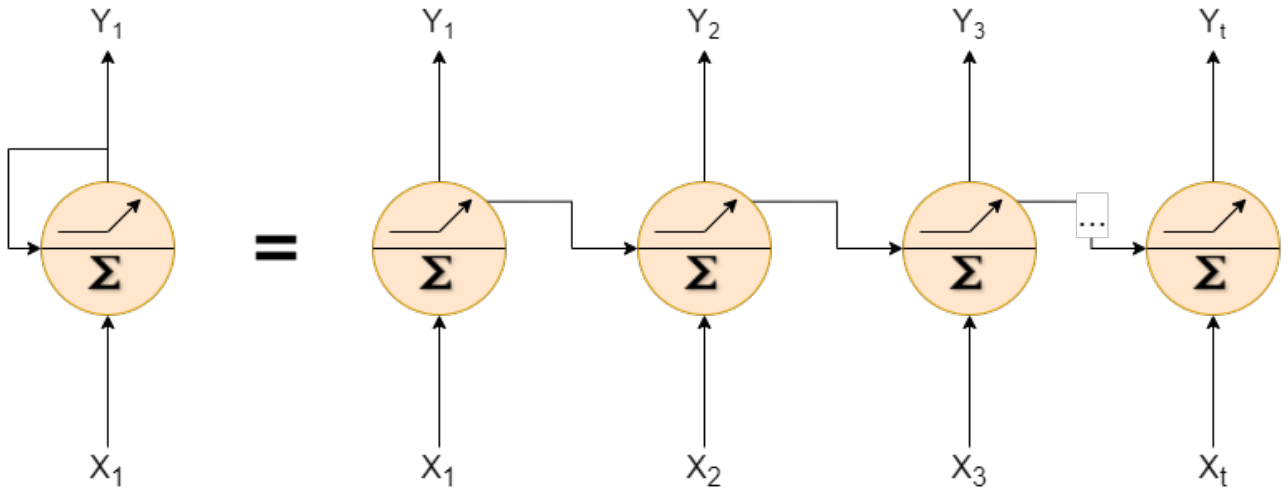

Figure 10: A simple Elman RNN using a ReLU activation function.

A limitation of this approach, is that the information being transferred cannot bypass a time step. We can therefore use Long Short-Term Memory units, which employ gates in order to control which data is used in the classification process at each time step[31]. The structure of an LSTM cell can be seen in Figure 11.

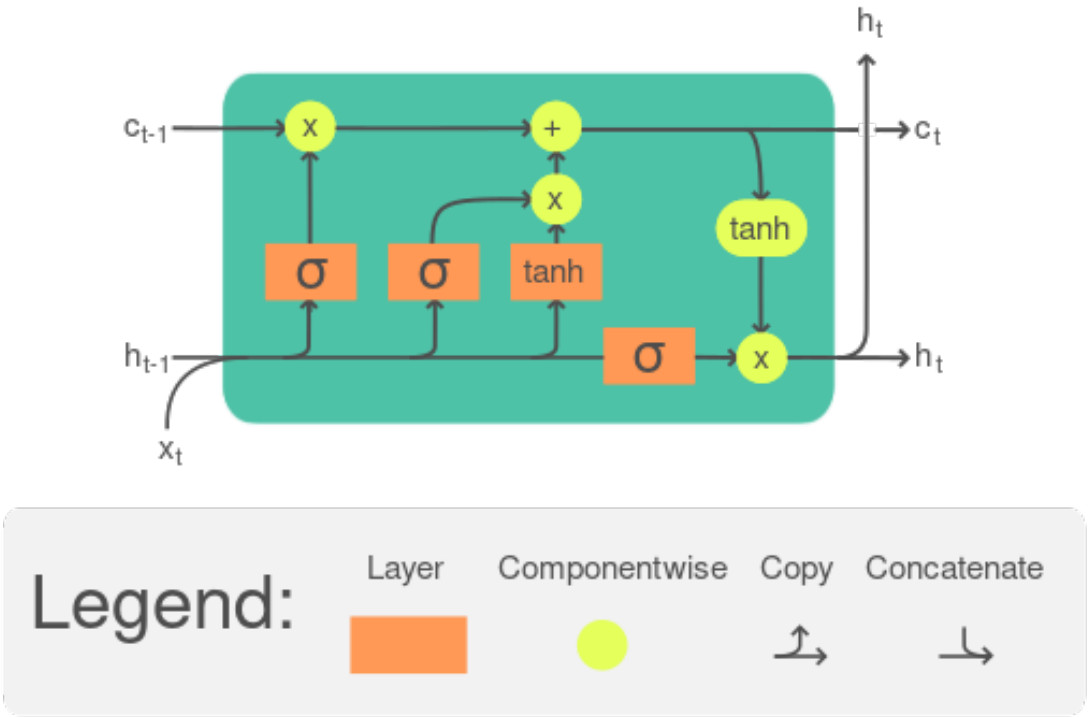

Figure 11: A diagram representing the structure of an LSTM cell. By Guillaume Chevalier - File:The_LSTM_Cell.svg, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=109362147

LSTM cells allow the flow of information over an extended period of time as they employ

three gates which allow them to control input, output, and the retention of previous data. The input gate controls whether the output of the current neuron is added to memory. The output gate controls whether the memory is passed along to the next time step. The forget gate controls whether the memory is emptied. Using a combination of these gates, LSTMs allow classification and prediction of time series data in which the output at time step t is reliant on only some of the previous time steps.

4.5.3 Technologies

The choice of technology to implement the neural network was fairly simple decision. As development would be in Python in order to facilitate fast development and use of the MediaPipe library, research was conducted into Python implementations of RNNs. It quickly became evident that the Keras library should be used, as there are a significant number of online resources detailing its RNN implementation[32]. It also provides a very modular approach to creating neural networks, meaning that new layers can easily be added, optimisation methods can be changed, and activation functions can be adjusted or replaced. As Keras is a wrapper for the TensorFlow library, it is also able to utilise features like TensorBoard for live tracking of accuracy and loss as the model is being trained[33].

A popular alternative to Keras for machine learning applications is PyTorch, which also has an implementation for Recurrent Neural Networks. Despite this, Keras will be used for this project, as the PyTorch implementation does not support adaptations of the ReLU activation function such as Leaky ReLU, and has much less documentation.

4.5.4 Neural Network Design

The initial design of the neural network is very simple, as it will be further developed and adapted during the iterative development process. The design is based on that used by Samaan et al.[15], as they claim it is capable of achieving 99% accuracy. Some changes will be made; for example, in the shape of the input, or the activation function used.

The initial design for the model can be seen in Figure 12, with the largest initial change being in the shape of the input data. The “None” in the shape of the input layer allows for the input of variable length data.

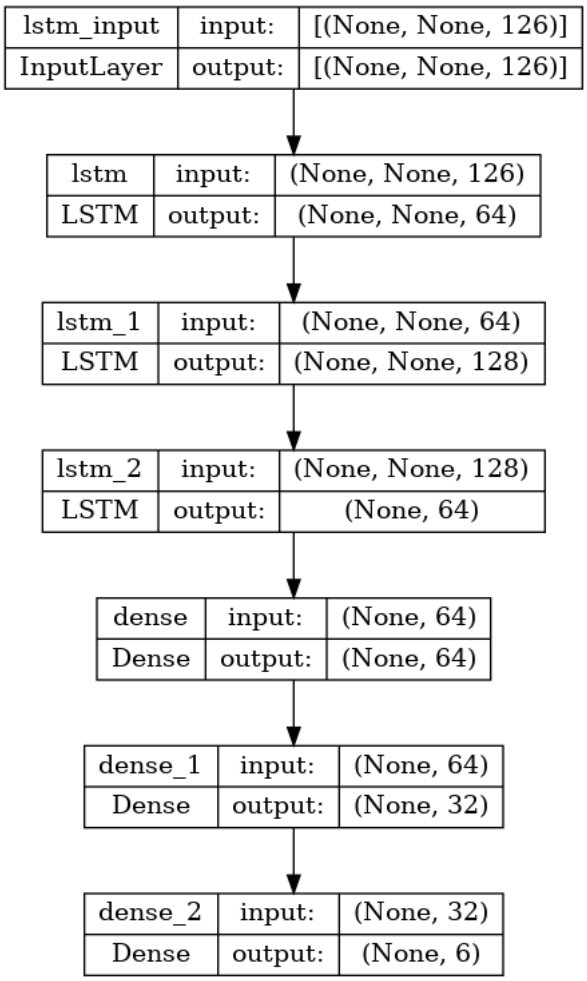

Figure 12: The initial design of the neural network.
Design adapted from *MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition*[15].

The activation function used by each layer is ReLU, as it solves the vanishing gradient problem which is present when the TanH function is used. If needed, ReLU can be replaced with Leaky ReLU or Parametric ReLU in order to solve the dying ReLU problem which can cause a network to cease learning[34]. Figure 13 demonstrates the difference between ReLU and Leaky ReLU.

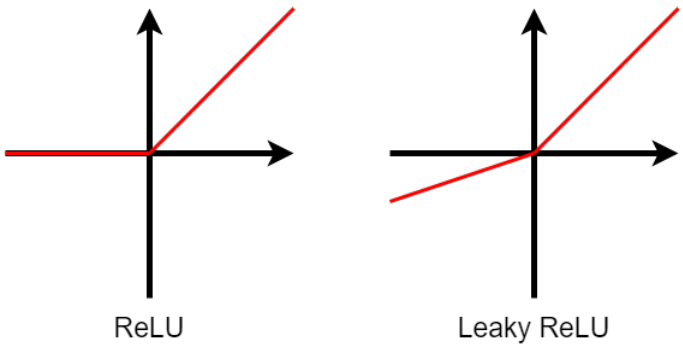

Figure 13: Depiction of the ReLU vs Leaky ReLU activation functions.

The Adam optimiser is used for this network, which is an implementation of stochastic gradient descent, as it is "computationally efficient" and "has little memory requirement"[35]. The loss function is categorical cross entropy, as the task is a multiclass problem where labels are one-hot encoded.

4.6 Testing

This section will define the testing procedure to be used during the iterative development stage of this project. Testing will be both quantitative and qualitative in order to give a fair assessment of both the quality of the proposed model, and its applicability to real world scenarios.

4.6.1 Quantitative Testing

Quantitative testing will be performed on a designated testing subset of the data on which the model has not been fit. This test set will be approximately 5% of the total data. Additionally, when the model is complete, some testing may be performed using the verified spottings which are present in the BOBSL dataset.

When the model is nearing completion, testing also incorporates k-fold cross validation. This will help to ensure that each of the splits accurately represents the dataset as a whole, which is not necessarily the case for a simple 95:5 split. By using K-fold cross validation, we can also calculate the average accuracy of the model with different training and testing datasets.

In order to analyse edge cases and the effects of certain sign characteristics, confusion matrices will be created after each test. In this way, we can easily see if two signs are frequently confused, or if changes to the system have a significant impact on the classification of specific signs.

4.6.2 Qualitative Testing

In addition to quantitatively measuring the quality of the model using test data, the system will be tested using a live video feed. This will allow manual experimentation with the model in order to determine areas of improvement and explore limitations with variations which may not be present in the dataset. Some such variations are the position of the signer in the frame, their distance from the camera and the speed of signing.

This testing will help inform about the quality of the dataset, the limitations of the model, and potential improvements to the user experience. It will also ensure that the proposed model works with a live video feed as well as pre-recorded videos, and with cameras whose frame rate or resolution may be different to that which is present in the dataset.

5 Data Acquisition

This section will detail the process of acquiring the video data, selecting the words to train the model on, and then extracting and storing the landmarks for each sample. The software developed in this section is fundamental to all stages of the iterative development process, and so it is important that the code is efficient and that design decisions are well considered.

5.1 Word Selection

As the spottings file contains over 18 million entries, there were a large variety of signs to choose from. A list was compiled of the words which appeared in the dataset with the highest frequency, as a large sample size is beneficial for the training of the model.

20 words were chosen, each of which has a minimum of 1000 instances. These words are shown in Table 1.

Word	**Samples**
hello	3679
you	58277
all-right	3801
some	14358
food	5247
want	11851
your	11563
family	4433
need	9215
house	5594

Word	**Samples**
come	11439
my	20927
old	4601
farm	4117
they	14733
would	15673
have-to	4932
whole	3316
horse	1603
eat	3319

Table 1: Table of the words selected for this project as well as their frequency in the dataset

The characteristics of these signs are varied. The signs for “all-right” and “house”, for example, use two hands; the signs for “food” and “eat” use the signers face; and signs such as “want” and “need” have large movements. These words also form some simple sentences, such as “Hello, you all-right?” and “Do you want some food?”.

A small selection of these signs will be used initially, with more signs being introduced later in the iterative process.

A short script was then created which takes this list of words, and for each word, generates lists of all the videos in which it appears, the timestamp within the video, and the probability of that instance being correct. These are then sorted using the probability as the key, such that the samples with the highest probability of being correct are chosen first. Any duplicates of the same word in the same video at the same time are removed. This data is then stored in a text file called *WordSpottings.txt*, ready to be used by the next component of the system.

5.2 Video Storage

As there are 1.6 TiB of data in the BOBSL dataset, the storage solution must be efficient, and capable of storing a large number of videos. As such, the project utilises 100GB of storage, capable of storing around 1000 videos. These videos can be downloaded automatically using the *wget* command and login details provided with access to the database.

The script responsible for downloading videos, cutting out the correct subclip, and then extracting and saving landmarks is called *GetDatapoints.py*. The main body of the script follows a simple algorithm, which is shown in Algorithm 1.

Algorithm 1 Downloading and Cutting Videos

```
words ← ["hello", "you", "all − right", "want"]
numSamples ← 200
samples ← readFile("WordSpottings.txt")
for word in words do
    wordSamples ← samples[word][: numSamples]
    for sample in wordSamples do
        if sampleVideo not downloaded then
            download video
        end if
        Cut subclip from video
        Run landmark detection
        storageUsed ← Calculate Storage Space Used
        if storageUsed ≥ 90GB then
            Delete 20 videos
        end if
    end for
end for
```

In the script, the word set and number of samples are defined manually, which are then used to retrieve the correct number of samples for the desired words from the *WordSpottings.txt* file. The program then loops through all of the samples and runs the aforementioned *wget* command using Python's OS library if the video file containing the clip has not already been downloaded. The script then uses the FFmpeg library[36] to cut out and save a short section of the video starting at the timestamp given in the sample. The initial length of the subclip is calculated using the framerate of the video such that the clip will contain exactly 8 frames. This can be easily adjusted in order to capture more frames. Landmark extraction is then run, which is described in Section 5.3.

The final step of the algorithm helps control the storage space used. The OS library is again used in order to determine how much of the 100GBs is currently filled. If the storage is more

than 90% full, the 20 least recently used videos are deleted. This will create enough storage space for approximately another 20 videos.

5.3 Landmark Extraction

The landmark extraction phase takes a short video and analyses each frame for hands. The implementation is based on that found in the MediaPipe documentation[13], with extra consideration for frames where there are no hands visible at all.

The algorithm for this section can be seen in Algorithm 2.

Algorithm 2 Landmark Extraction

```
video ← vid.mp4
allLandmarks ← {}
for frame in video do
    landmarks ← detectHandLandmarks(frame)
    if both hands detected then
        lHand ← landmarks.left
        rHand ← landmarks.right
    else if left hand detected then
        lHand ← landmarks.left
        if rHand is NULL then return
        end if
    else if right hand detected then
        rHand ← landmarks.right
        if lHand is NULL then return
        end if
    else
        if lHand is NULL or rHand is NULL then return
        end if
    end if
    allLandmarks.append(lHand, rHand)
end for
save allLandmarks
```

In this procedure, the video is opened using the Python OpenCV library[37], which allows us to iterate through each frame in the video. The MediaPipe hand landmark detection model is then applied to each frame. The initial implementation of this algorithm exits from the function if any hand which is not present in the image has no previously known locations. In effect, this means that if both hands are not found in the first frame of the clip, the sample is discarded completely. The two reasons for this approach are simplicity and concerns over data

quality. Through initial testing of this algorithm, it was found that some samples referenced video clips similar to that seen in Figure 14, in which not only is the label incorrect, but there was no signer in the frame. Therefore, these mislabelled samples can be discarded. If the last known location of a missing hand is known, then these landmarks are used. Once the landmarks from all 8 frames have been collected, the data is saved.

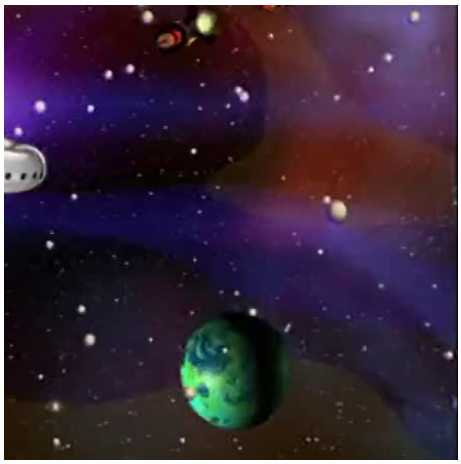

Figure 14: Frame from a video clip labelled "hello"

This algorithm can be easily changed during the development process if the format of the data needs to change. For example, Section 6.6 sees this algorithm changed to allow for missing hands in up to the first 6 frames, and extends the total number of frames analysed to 24. This algorithm could also be adapted with relative ease to use hand trajectory to estimate the position of each landmark if a hand is not found, rather than using the last known location. This could assist with videos in which there is motion blur, as the hand position would continue to move rather than appearing stationary.

5.4 Data Formatting

As the landmark detection model returns 42 landmarks, each with 3-dimensional coordinates, there are 126 features per frame. Their default formatting is in MediaPipe-specific data structures, which are not compatible with the Keras neural networks. Therefore, before the data is saved, the landmarks are converted to a Numpy array[38]. Each array takes the form $lmks = [l_{x1}, l_{y1}, l_{z1}, \ldots, l_{x21}, l_{y21}, l_{z21}, r_{x1}, r_{y1}, r_{z1}, \ldots, r_{x21}, r_{y21}, r_{z21}]$.

Once the data is gathered, it must be saved so that it can be used for training the neural network. Numpy supports the saving and loading of arrays directly, meaning that no other data manipulation is required for this process. The data for each frame is saved in a file structure which follows the format seen in Figure 15.

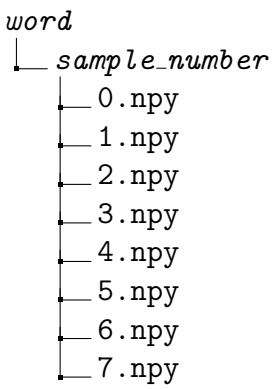

Figure 15: The directory structure for saving landmark data.

As the data is stored as Numpy arrays, a simple script was written to iterate through the directory structure and load the each sample. This means that this approach is easily expandable to more words, more samples, and more frames, ensuring compatibility throughout the development process. It also allows simple analysis of the data if required, as individual frames can be analysed by opening the corresponding file.

The use of Numpy arrays also allows for efficient mathematical operations over the entire dataset if necessary, for example to find duplicates, outliers, or similarities. It should be noted, however, that Numpy does not support 2-dimensional arrays where the sub-arrays are of different lengths. Therefore, when variable length data is introduced (see Section 6.6), the frame data is stored in a ragged Tensor, a Tensorflow data structure designed for variable length sub-lists[32].

6 Iterative Development and Testing

This section details the iterative development process. We first describe the development process for the live testing script. After this, we detail the process of training the model and analysing its performance using both quantitative and qualitative techniques, as described in Section 4.6. We then summarise the accuracy of the models developed, their limitations, and the problems encountered during development. Each subsection between 6.2 and 6.8 covers a new iteration.

6.1 Live Testing

The first program created in this phase of development was a script for live testing. This is a conceptually simple program, which uses OpenCV to capture a live video feed from a camera, processing every frame almost exactly as described in Algorithm 2. The key difference is that instead of exiting if no hands are found, the program simply continues to the next iteration of the loop. This program is based loosely on a similar open source project by Nicholas Renotte[39].

After each frame's data is processed and formatted (see Section 5.4), it is appended to a 2-D array containing landmark data from previous frames. This array is then shortened to the most recent 8 frames, which can be passed directly to the model for classification.

As the output layer of the model applies SoftMax, each word is given a probability such that their sum is equal to 1. The maximum of these is the assigned class. We only accept a classification if the probability is above some threshold value, initially defined to be 0.95. We then also reject the classification if it is the same as the last accepted classification. This is to avoid instances where frames 0 - 7 and 1 - 8 are given the same classification, resulting in duplicate words in the output.

The final section of the live testing script displays the feed from the camera with the location of the landmarks overlaid. It also displays the last 5 words which have been classified and accepted. This allows us to see in real time which landmarks are used for classification, as well as their output. We can therefore see misclassifications and attempt to identify which movements they are caused by. Figure 16 demonstrates the script in action, recognising and outputting the sign for "hello" in difficult back-lit conditions, and with hand landmarks overlaid onto the video.

Figure 16: A demonstration of the live testing script recognising the sign for "hello".

This script can easily be adapted as changes are made to the system throughout development. As the majority of the algorithm is based on the implementation for gathering data from videos, any changes to one can be extended to the other. Additionally, changes to the model are largely inconsequential, as the model being loaded by this script can be easily changed. If the word set changes, then the word set this script is using must also be updated.

6.2 4 Words - 200 Samples

6.2.1 Data

The initial data for testing uses a small selection of words, which are "hello", "you", "all-right", and "want". These were selected due to the variation in sign, as well as the presence of a two-handed sign and one involving lots of movement. The sample size for each word is 200, meaning that the 200 instances of each word with the highest probability of having correct labels are chosen. Each sample contains 8 frames.

First, the data gathering script was run to download the videos, cut out the sample clip, and extract hand landmarks. This resulted in several hundred samples being discarded, which could indicate several things:

1. The dataset contains many samples which do not contain signers.
2. The dataset contains many samples whose first frame contains motion blur.
3. The dataset contains many samples in which one or both of the signer's hands are occluded in the first frame.

As the focus of this iteration is on creating a minimum viable product, these possibilities are explored in future iterations.

There are approximately 620 samples, and a train-test split of 95:5 is used, resulting in the test set containing approximately 31 samples. This split is generated at random using the Scikit Learn library[40]. Intuitively, the training set is used to train the model, and the testing set is used for quantitative testing including accuracy and the generation confusion matrices, as described in Section 4.6.

6.2.2 Model

The neural network implemented in this iteration is as seen in Figure 12. There are three LSTM layers, and three dense layers, as proposed by Samaan et al.[15]. The optimiser used is Adam, as this is a standard stochastic gradient descent implementation, and the loss is categorical cross entropy. The code for this can be seen in Figure 17.

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu',
↪  input_shape=(None, 126)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(optimizer='Adam', loss='categorical_crossentropy',
↪  metrics=['categorical_accuracy'])
model.fit(x_train, y_train, epochs=500, callbacks=[tb_callback])
```

Figure 17: The implementation of the neural network in Python using the Keras library

The model is trained for 500 epochs, and the TensorBoard tool included in TensorFlow is used to create graphs of the accuracy and loss each epoch[33]. This allows analysis of accuracy over time in order to better tune the hyperparameters in future iterations and ensure the model is not under or overfitting. This will also allow us to view limitations of the activation function used, such as dying ReLU, which is discussed further in Section 6.6.

6.2.3 Quantitative Results

Using TensorBoard, the graph of accuracy over epochs was created, which can be seen in Figure 18.

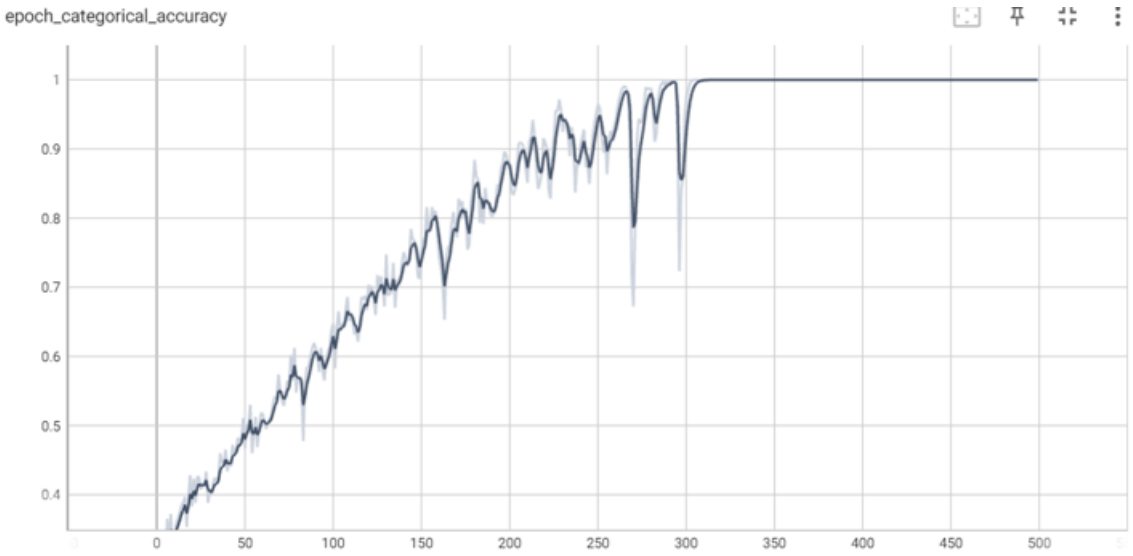

Figure 18: The accuracy of the model over 500 epochs for 4 words.

As this graph plateaus at 100% accuracy, we know that the model is overfitting to the training data. Therefore, the number of epochs should be reduced if the same data is used for future iterations. If more words or samples are used, then the number of epochs before the model is overfitting may increase, and so this hyperparameter varies each iteration in order to achieve the best results from the model.

The model was then tested on the test data, which resulting in a test accuracy of 52%. This result is twice that expected from random guessing, indicating that the model is capable of classifying signs with some accuracy. There may be several reasons that this is not more accurate, but the most likely reason is that the model is overfit on the training data. Another cause might be that the training data is not as clean as previously thought, and contains samples which are incorrectly labelled. This is a consideration for future iterations.

Using the test data, confusion matrices were created for each word. This will allow us to easily identify signs which are frequently misclassified, and in later iterations, whether similar signs are often confused for each other. The confusion matrices for this test can be seen in Figure 19. “TN” refers to True Negative, “FP” refers to False Positive, “FN” refers to False Negative and “TP” refers to True Positive.

Figure 19: Confusion matrices for 4 words.

In an ideal situation, all values would be along the shaded diagonal, with zero in the bottom left and upper right quadrants. This would indicate that all test samples were correctly classified. From these confusion matrices, we can see that "hello" and "want" are classified correctly much more often than "all-right" and "you". This likely indicates problems within the dataset, which is addressed in Section 6.3.4.

6.2.4 Qualitative Results

This model was tested using the live testing script previously described in Section 6.1. Using this model, we were able to verify that the script performs as expected, applying the hand landmark detection model to each frame in a live video, passing this data to the neural network, classifying the sign, and displaying the last 5 classifications on the screen.

After some simple testing, it is clear that the model behaves as expected in this setting, and is able to recognise about half of the signs. It is also clear that accuracy improves when variables similar to the test data are emulated, with the signer at a fixed distance from the camera and central in frame. More detailed qualitative testing is reserved for future iterations which use more words and samples, where we can gain further insight into the limitations of the model or dataset.

6.3 6 Words - 400 Samples

6.3.1 Changes

This iteration involved one major change, which was the introduction of two new signs into the word set. These signs are "some" and "food", which were chosen as they both involve a single hand in a similar position and orientation. The sign for "some" is a snapping motion of the thumb against the forefinger whilst the hand moves slightly forwards, and the sign for "food" is the thumb touching the tips of all fingers tapping against the signer's mouth twice.

As these two signs both include a single hand, which is bunched and with the back of the hand facing the camera, we can anticipate a reduction in accuracy, and particular confusion of these two signs.

The sample size for each word is also increased over the previous iteration, from 200 to 400 samples per word. This leads to a maximum of 1600 samples, although due to the data extraction process, the final number is closer to 1000. This change was anticipated to have little effect on the model, as the introduction of two new signs will likely decrease the accuracy more than the larger sample size will increase it.

The focus of this iteration is on observing the effect of introducing new words into the word set, and observing whether the system is capable of differentiating signs which have some similarities. The model otherwise remains the same for this test in order to isolate the effects of changes.

6.3.2 Quantitative Results

The quantitative results for this iteration are as expected based on the findings of the first iteration. During the training stage, the model still overfits due to the large number of epochs. The graph of accuracy over epochs can be seen in Figure 20.

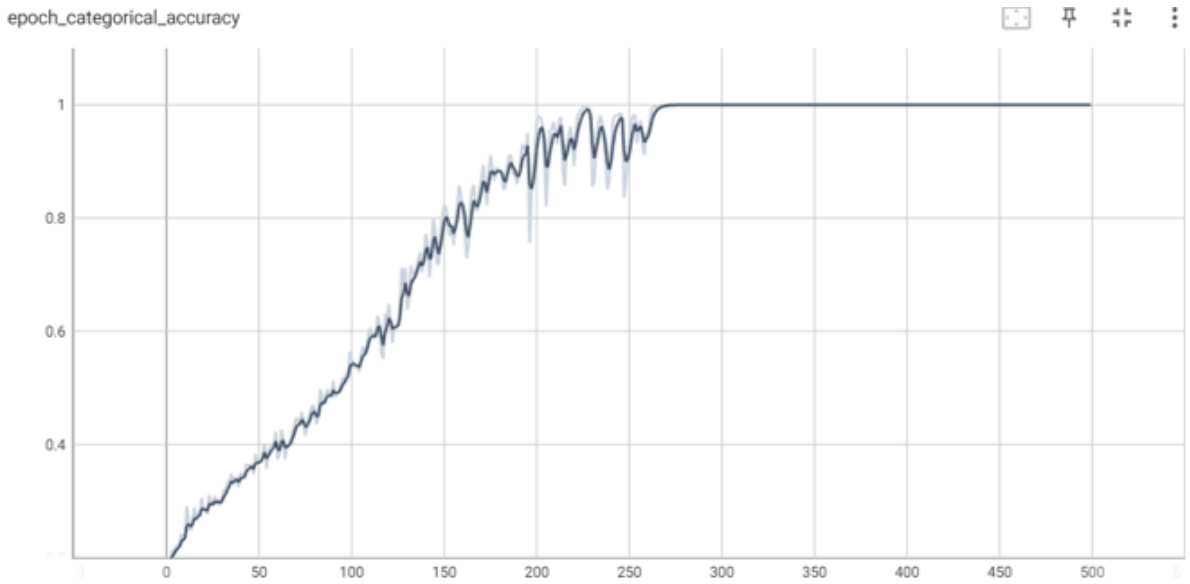

Figure 20: The accuracy of the model over 500 epochs for 6 words.

As the model is overfit, and there are a larger number of classes, the accuracy of the model on the test set is reduced to 36%. This is again slightly greater than double the expected accuracy of random guessing, which would be $16.\dot{6}$, although not close to the results achieved by Samaan et al.[15] using a very similar model, albeit including pose data and with a custom dataset. As the signs chosen do not rely largely on movement of the body, and involve more hand movement, it seems likely that the data available is restricting the capability of the model, as well as the number of epochs causing overfitting, which is adjusted for future iterations.

The confusion matrices for this test set were then generated in order to analyse the model's classification capabilities on the two newly introduced signs. These matrices can be seen in Figures 21 and 22.

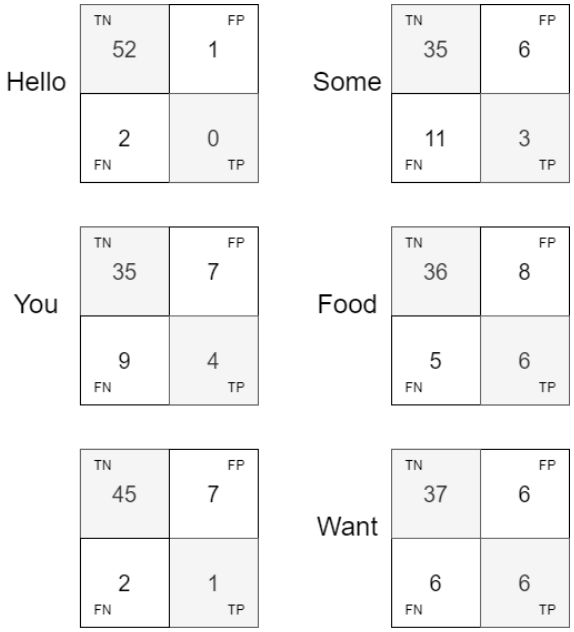

Figure 21: Confusion Matrices for 6 words.

		Predicted					
		Hello	You	All-right	Some	Food	Want
Actual	Hello	0	0	1	0	0	1
	You	0	4	3	2	2	2
	All-Right	0	1	1	0	1	0
	Some	0	4	1	3	5	1
	Food	0	0	0	3	6	2
	Want	1	2	2	1	0	6

Figure 22: The detailed confusion matrix for 6 words.

These confusion matrices tell us several things. Firstly, "some" and "food" are frequently confused, as we had previously predicted. The confusion between these two signs will be monitored and analysed as the model develops. Secondly, the sign for "you" is frequently misclassified, and is commonly used as the incorrect label for "some". This second point could be due to the similar hand shapes involved in the signs, however in conjunction with the many misclassifications of "you", likely indicates underlying issues with the data. Therefore, further analysis of the data was performed prior to the development of the next iteration.

6.3.3 Qualitative Results

When performing live testing, it was discovered that this model gave a huge number of false positives, which is expected based on the results in Figure 21. This meant that it was effectively impossible to test for other features, such as performing similar signs, and varying position in the frame or signer speed. Many of these false positives arose from classification of non-sign frames, in which the signer's hands were stationary, moving between signs, or moving in other ways which were not signs.

This large number of false positives further indicates underlying problems with the data which were not detected during initial exploration of the dataset.

6.3.4 Further Dataset Analysis

As the dataset used for training is in the form of NumPy arrays, it is impossible to interpret directly. As such, a Python notebook was created which reads all of the data for each sample, and displays each of the frames in row.

As the method provided by MediaPipe for displaying the landmarks on an image requires the MediaPipe landmark classes, whose constructors are not exposed to the user, simple classes were created to mimic them. These classes implement the bare minimum required for the landmark drawing function and are based on the open source code for MediaPipe.

```
class landmark:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def HasField(self, _):
        return False

class landmarkList:
    def __init__(self, landmarks):
        self.landmark = landmarks
```

Each sample is displayed in a row and labelled as it is in the dataset. This means we are able to analyse the movement of each landmark across the frames and manually determine whether the label is correct. Due to the large number of samples, manual analysis of frames is computationally faster than compiling the frames into videos, and also has the benefit of more easily comparing samples.

Through this analysis, it quickly became clear that there were significant issues with the labelling of the data. While there were some samples that were correctly labelled, there appeared to be many more which were incorrect. For example, Figure 23 depicts a sample of the word “hello”. The sign, as one might expect, is the waving of a hand; however, these frame images clearly demonstrate that the signer in the clip does not move their hands from in front of their stomach.

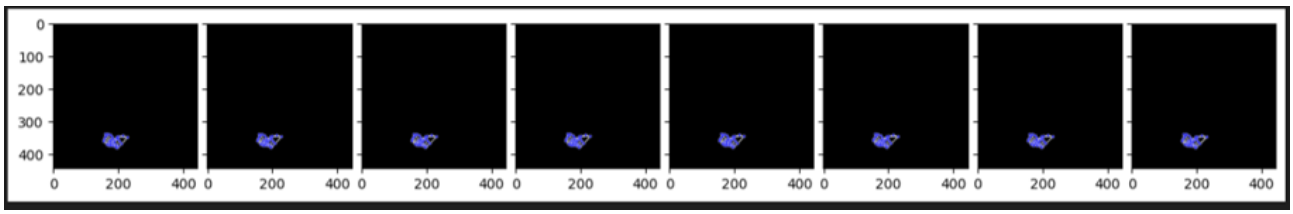

Figure 23: The frame-by-frame representation of a sample for the word "hello".

This mislabelling of data is persistent throughout the 6 different classes. The most likely cause of this is the automatic annotation algorithm employed in the generation of the dataset[22]. The accuracy of the techniques used to perform the annotation is approximately 75%, although that may vary depending on the signs in question. It may also be the case that the timings associated with each sample are inaccurate.

Using this information, the following iterations will focus on adjusting the data which is used in order to maximise the potential of the model.

6.4 6 Words - Disproportional Clean Data

6.4.1 Changes

Due to the disproportionate data used for training and testing of the model, the results obtained from this iteration serve little purpose in the consideration of the system's performance. However, it does serve as the foundation of future iterations and was an important learning opportunity in the development process.

The first step taken in improving the quality of the data was the relabelling of all current samples. As such, a short program was written which iterated through each sample, showing the current label and frame image representation, allowing us to either accept the current label or relabel it as a new class called "unrecognised_sign".

After manually cleaning the data, around 700 samples were considered unrecognised, whilst each of the six classes for signs had between 30 and 60 verified instances. This significant discrepancy indicates that either the sign is not present in the 8 frames after the time stamp given in the majority of cases, or that the automatic labelling system has an accuracy of roughly 30% for these words, which is very unlikely.

Although the sample size is heavily skewed towards the "unrecognised_sign" class, this model should provide some insight into the benefit of having a dedicated class for non-signs, which is a technique applied in some previous works[6, 10]. A non-sign class may be applicable when a signer's hands are moving between signs, stationary, or performing other non-sign movements.

No changes were made to the model in this iteration, as the focus is on analysing the effect of relabelling the data.

6.4.2 Quantitative Results

Due to the disproportionate sample sizes between classes, we expect this model to train very quickly to recognise "unrecognised_sign" samples. This would result in 70% accuracy from always assigning this label regardless of input. The accuracy of the model can be seen in Figure 24.

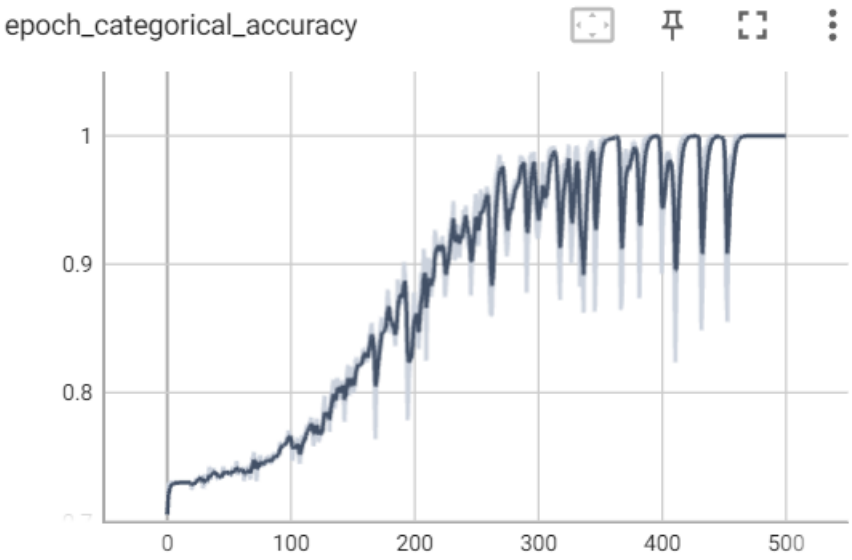

Figure 24: The accuracy of the model over 500 epochs for 6 words using clean data.

It should be noted that this graph's accuracy starts at 70%, proving the hypothesis that the model quickly learns to classify "unrecognised_sign" instances. The test data further supports this theory, achieving an accuracy of 66%. Although this seems like an improvement over previous iterations, the accuracy is misleading due to heavy favouring of the non-sign class. This may be further exacerbated by the sample size of each of the signs, which is low irrespective of the number of unclassified sign instances.

The confusion matrices also lead to the same conclusion. As shown in Figure 25, the "unrecognised_sign" class is represented far more significantly than any other class. This makes it difficult to analyse the accuracy of the model or confusion between signs.

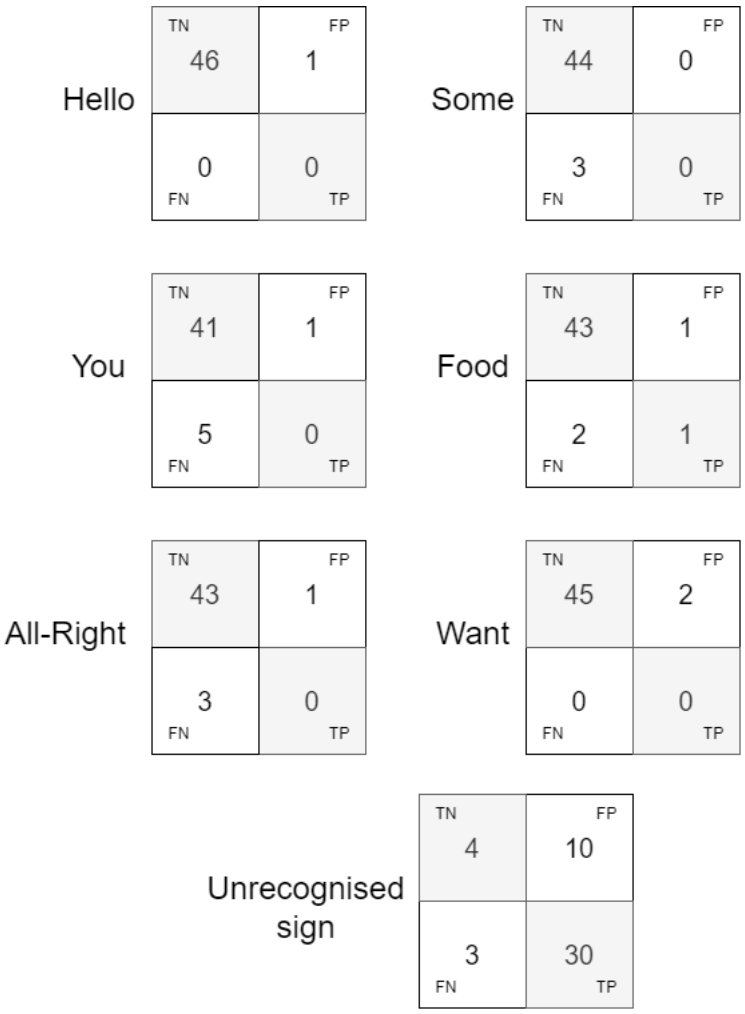

Figure 25: Confusion matrices for 6 words using clean data

6.4.3 Qualitative Results

Qualitative analysis using the live testing script revealed no new information about the model. Although there was no longer a significant number of false positives which had been present in previous iterations, there were now far more false negatives for actual signs. Therefore, the accuracy of the model is not representative of its ability to recognise signs, but rather representative of its ability to recognise non-signs. As outlined in Section 6.4.1, the results gathered by this iteration are not considered in the evaluation of the project.

6.5 6 Words - Proportional Clean Data

6.5.1 Changes

Based on the findings of the previous iteration, this iteration fixes both the problem of the disproportionate data and the model overfitting. As the number of samples for each of the sign classes ranges from 30 to 60, we change the training algorithm slightly to only select 60 random samples from the "unrecognised_sign" class. This means that the training and test

data contains an evenly distributed number of samples for each class.

Due to the now much lower number of samples, this iteration uses 10-fold cross validation in order to ensure that the accuracy score is representative of testing over the entire dataset. The number of epochs is also reduced to 200 from 500 in order to prevent overfitting.

The focus of this iteration is estimating the effectiveness of introducing a class which represents unrecognised movement of the hands, and so no changes were made to the structure of the model.

6.5.2 Quantitative Results

As the number of epochs are reduced, the accuracy graph no longer displays a plateau at 100% accuracy for the training data. This indicates that the model is no longer overfitting, and thus should achieve better accuracy on the test data. Figure 26 depicts one run from the 10-fold cross validation, which is representative of the average accuracy achieved on the test data set.

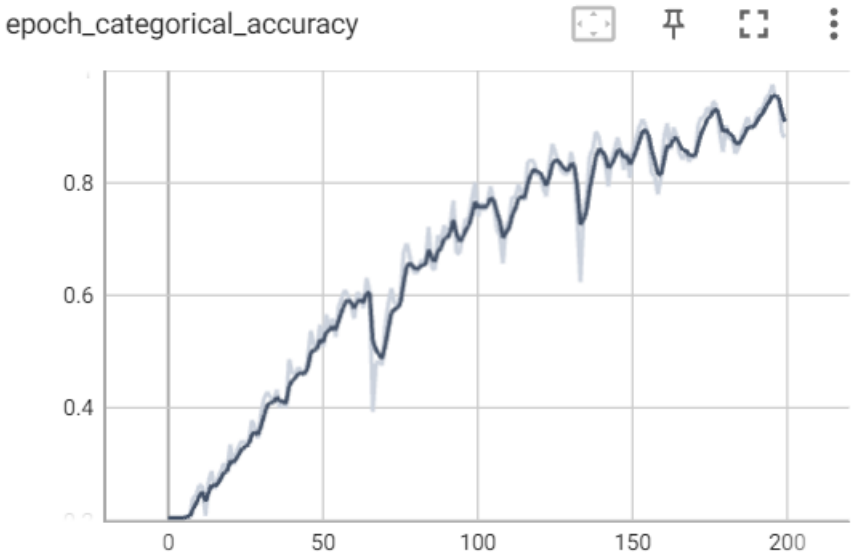

Figure 26: The accuracy of the model over 200 epochs for 6 words and unrecognised sign using clean data.

The average accuracy over 10 runs of this model is 50.8%, which is notably better than in previous iterations. Additionally, as there are now seven classes, the estimated accuracy for random guessing is 14.2%, which means that this model is approximately 3.5 times more accurate than selecting a class at random. This increased accuracy is likely due to the lack of noise in the data, particularly samples which appeared to be present in all classes, causing previous models to struggle with differentiating between them.

As the sample size is relatively low, the confusion matrices are largely inconclusive with respect to which signs are frequently confused. However, they do highlight that this model particularly struggles with the sign for “you”. This can be seen in Figure 28, and is likely due in part to the nature of the unrecognised signs, many of which contain arbitrary motions of

a single hand moving between signs. In a short clip of just 8 frames, the sign for "you" can therefore be easily confused with such motions. This, however, does not explain the confusion between "you" and "all-right", the cause of which we are unable to determine.

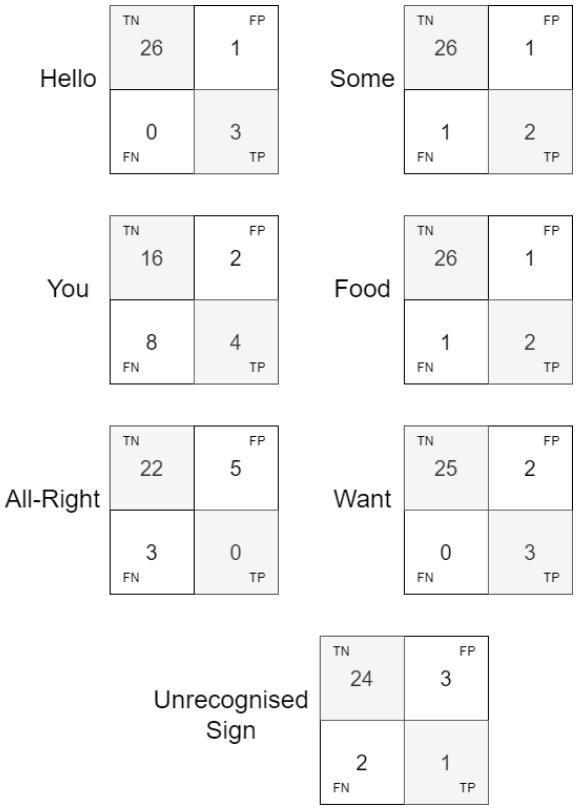

Figure 27: Confusion matrices for 6 words with clean and proportional data.

		Predicted						
		Hello	You	All-right	Some	Food	Want	Unrecognised Sign
Actual	Hello	3	0	0	0	0	0	0
	You	1	4	4	0	0	1	2
	All-Right	0	1	0	1	0	0	1
	Some	0	0	0	2	1	0	0
	Food	0	0	1	0	2	0	0
	Want	0	0	0	0	0	3	0
	Unrecognised Sign	0	1	0	0	0	1	1

Figure 28: The detailed confusion matrix for 6 words with clean and proportional data.

6.5.3 Qualitative Results

In live testing, this model performs more reliably than any previous iterations. It is capable of recognising when no sign is being performed, as well as recognise with some accuracy most signs. As the confusion matrices suggest, it struggles with the signs "you" and "all-right", although it also frequently confuses the signs for "hello" and "food". This is potentially due to similarity in the movement of the arm, from in front of the body upwards, which takes the majority of the 8 frames available.

Despite the improvement in classification accuracy, it is clear that this model is still heavily reliant on features present in the training data, such as the signer's distance from the camera and the speed of their signing. Signs such as "hello" appear to be recognised more frequently when the motion of the hand is in the upper left of the frame, as the signer's hands are in the original dataset. This indicates that the movement of the hand has less of an impact on the classification than expected, and more weight is given to its position in the frame.

6.6 6 Words - Verified Data, 24 Frames

6.6.1 Changes

Despite promising progress made in the previous iteration, it was decided that development should take a slightly different track. This is because the most impactful way of improving the previous model would be by increasing the number of samples for each sign, which is unfortunately not feasible due to the time required to relabel the data. Instead, we now explore the other potential obstacle discussed in Section 6.3.4, which is that the timings assigned to signs in the original dataset are inaccurate to some degree. If this is the case, then some samples, which were previously considered unrecognised, may contain the sign of their label after the first 8 frames. Therefore, considering more frames for each sample will reduce the number of unrecognised cases.

As the videos contained in the dataset are 24 frames per second, we adapt our landmark extraction algorithm in order to capture 24 frames worth of landmarks. Therefore, the data for each sample now corresponds to one second of video. We further adapt this algorithm to allow for no hands in up to the first 6 frames. If, by the sixth frame, both hands have not yet been discovered in the video, then the sample is discarded. This provides several benefits over the previous approach. Firstly, if the hands are not present in the first frame, due to motion blur from movement or occlusion of the hands by each other or other parts of the body, then the sample is no longer discarded. Secondly, this allows us to further utilise the power of the recurrent neural network in processing data which is of variable lengths. With this new system, if both hands are not present in the first 3 frames, but are in the fourth, then the sample will contain 21 frames. The minimum length of a sample is therefore 18 frames, and the maximum is 24.

The aim of this is two-fold. Firstly, it will mean that if the sign is present, it will be captured and represented. Secondly, there is more data available to the model which it can use to help differentiate between signs which have similarities, such as “hello” and “food” or “food” and “some”.

Additionally, we adjust the source of the data for this task. Instead of using the large spottings file, we use the smaller, verified version. This file contains the same information about each word: video name, timestamp, and probability of correctness; however, each sample has been verified by a Deaf signer[22]. This means that the data should be clean and free of mislabelled samples without having to manually relabel it. This does, however, mean that a new set of words must be used, as the words used so far are not prevalent in the verified set. The same analysis of information was performed as described in 5.1. This revealed that over 500 of the 761 words in this verified dataset had fewer than 10 verified instances, which is due to the random sampling method used for verification. As such, 6 words with over 250 samples each were selected to be used for this iteration. These words are shown in Table 2.

Word	Samples
good	611
first	552
feel	468
family	338
look	283
house	277

Table 2: Table of the verified words selected for this iteration, and their frequency.

As it happens, these signs share some similarities with the signs used in previous iterations. "Good" and "all-right" both use thumbs-up, although the former with only one hand and the latter with 2. The sign for "first" is relatively similar to that for "some", as a single hand with its back to the camera in front of the signers chest is used for both. The sign for "look" is also somewhat similar to "food" from the perspective of a camera, requiring the signer to touch their face, and the sign for "house" uses two hands, moving to form the outline of a house. Therefore, this selection of signs will allow testing of the same sign characteristics previously analysed.

Another change made in this iteration is to the activation function used in the neural network. During training, it was noticed that the accuracy would sometimes instantly drop by up to 50%, and either recover slowly or not at all. The cause of this is Dying ReLU[34], which means that a number of nodes used in the learning process encountered values which were less than or equal to zero, and thus they were no longer able to learn. This was fixed by replacing the activation function with Leaky ReLU, an adaptation of ReLU which, instead of outputting zero for negative values, multiplies the input by some small constant α. We use a value of $\alpha = 0.3$. This means that all nodes in the model are able to continue learning.

As the model was already configured to accept inputs of variable length, no other changes are made to its design, although the number of epochs used does change from previous iterations in order to prevent overfitting. The updated diagram can be seen in Figure 29. Note that Leaky ReLU appears as its own layer due to Kera's implementation.

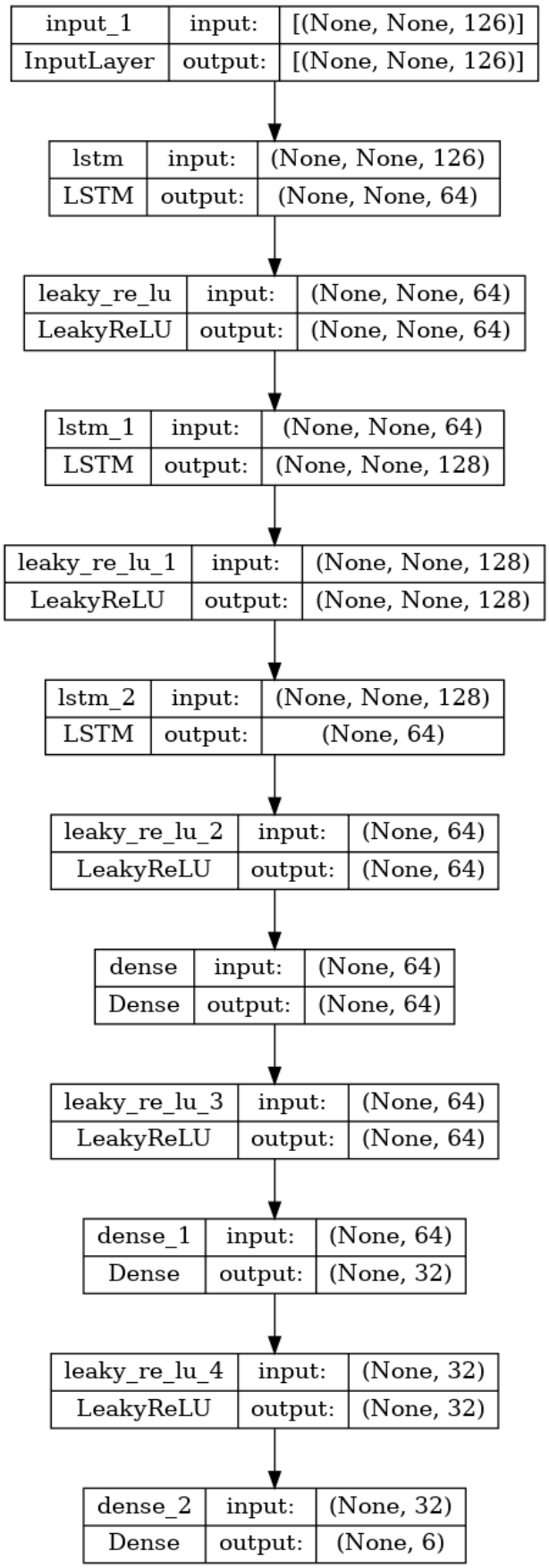

Figure 29: The updated version of the model, which utilises Leaky ReLU.

6.6.2 Quantitative Results

This model was tested using 10-fold cross validation in order to ensure that the accuracy recorded is representative of the average performance. 75 epochs are used for this testing, as the model overfits more quickly than in previous iterations, likely due to there being much more data available to use for classification. Figure 30 demonstrates the accuracy over the epochs of the model on the training data.

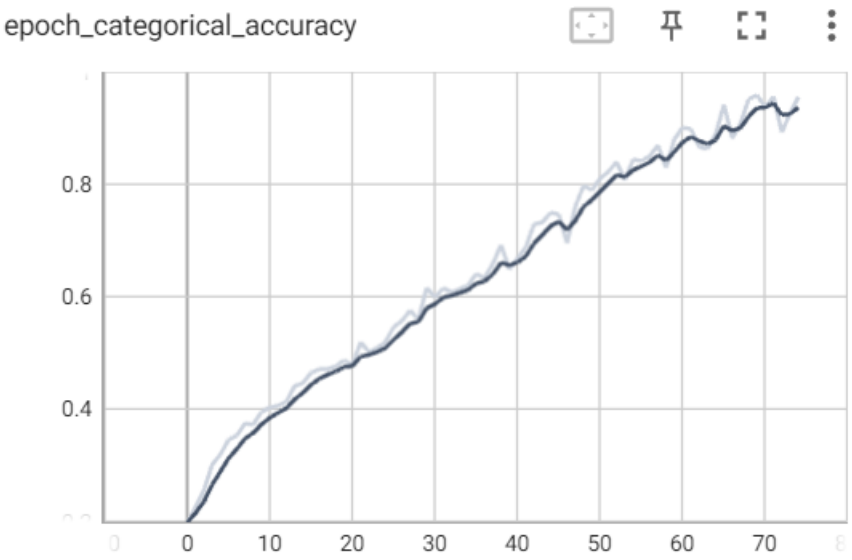

Figure 30: The accuracy of the model over 75 epochs for 6 words from the verified dataset.

The average accuracy achieved is 55.7%, which is lower than we would expect given that there should be little or no noise in the dataset, and that each sign is performed in a very similar manner in every sample. The quality of the data should be relatively similar to that used by Samaan et al.[15], as many variables are close to constant, such as the framerate, distance of the signer from the camera, the speed of the signer, and the position of the signer in the frame. A similar neural network architecture is also used, and up to 5 times as many samples per sign. They used only 75 clips for each sign, whereas this model is trained on up to 370 for each. Despite these similarities, the accuracy is barely half of that proposed by their solution, suggesting that there are other factors at play.

The confusion matrices for one fold's testing are shown in Figures 31 and 32. From these, we can see that the model has a high accuracy on some words, yet struggles with others. Notably, "house" is classified correctly every time, with no false negatives, and only 5 false positives. It seems likely that this high accuracy is due to the fact that it is the only sign in this set which involves both hands. The false positives may be caused by samples in which both hands are involved in movement, either due to the signer returning their hands to their stomach from a previous sign, or due to mislabelling.

We also note that there is a very high level of confusion between the signs for "look" and "first". This is again likely due to similarities in the movement involved in each. The sign for

"look" involves moving the extended index finger and middle finger up to the eye, and then away in a diagonal movement. The sign for "first" involves holding the index finger up in front of the signer's chest, with the palm facing away from them, before rotating at the wrist so that the palm faces the signer. Both of these signs involve moving the hand upwards from the stomach, and with similar hand shapes. This could explain why these two signs are frequently confused.

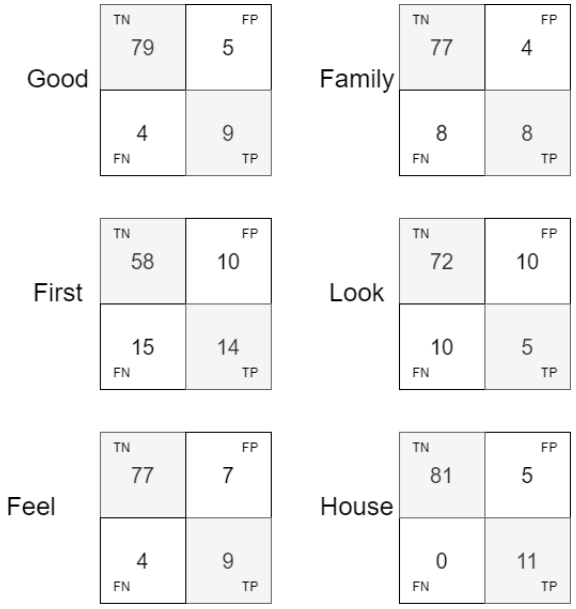

Figure 31: Confusion matrices for 6 words from the verified dataset.

Actual \ Predicted	Good	First	Feel	Family	Look	House
Good	9	0	2	0	1	1
First	2	14	2	2	8	1
Feel	1	2	9	1	0	0
Family	0	3	3	8	1	1
Look	2	5	0	1	5	2
House	0	0	0	0	0	11

Figure 32: The detailed confusion matrix for 6 words from the verified dataset.

6.6.3 Qualitative Results

The performance in live detection is much as expected based on the quantitative results. The limitations present in previous models such as the signer's position in frame and distance from the camera are also still present, although sign classification is improved over previous iterations.

The sign for "house" was recognised almost every time, and the signs for "look" and "first" are commonly misclassified. Despite this, the model is a noticeable improvement over previous iterations for recognition of multiple sequential signs, besides that developed by the previous iteration. Using this model, there are few false positives; however, they do occur when the signers hands are moving out of a previous sign. This is likely a result of the lack of a class dedicated to "unrecognised_sign", which was present in the previous iteration.

Additionally, the true positive rate for the signs "good", "family" and "feel" is noticeably low, although not as low as "first" and "look". As this is reflected in the data gathered from qualitative testing, it could indicate that there are still some issues with the data on which the model is trained. Therefore, the same data analysis techniques were employed, this time representing the 24 frames of data on three rows of 8 frames each.

Analysing this data revealed several things. Firstly, the suspicion of inconsistent timings was proved correct. In some samples, the sign occurred in the first 8 frames, whereas in others, the sign was not present until halfway through the sample. This misalignment could be the cause of the model's relatively low accuracy. In the extreme case, there are some samples whose sign is not present at all, for example that seen in Figure 33.

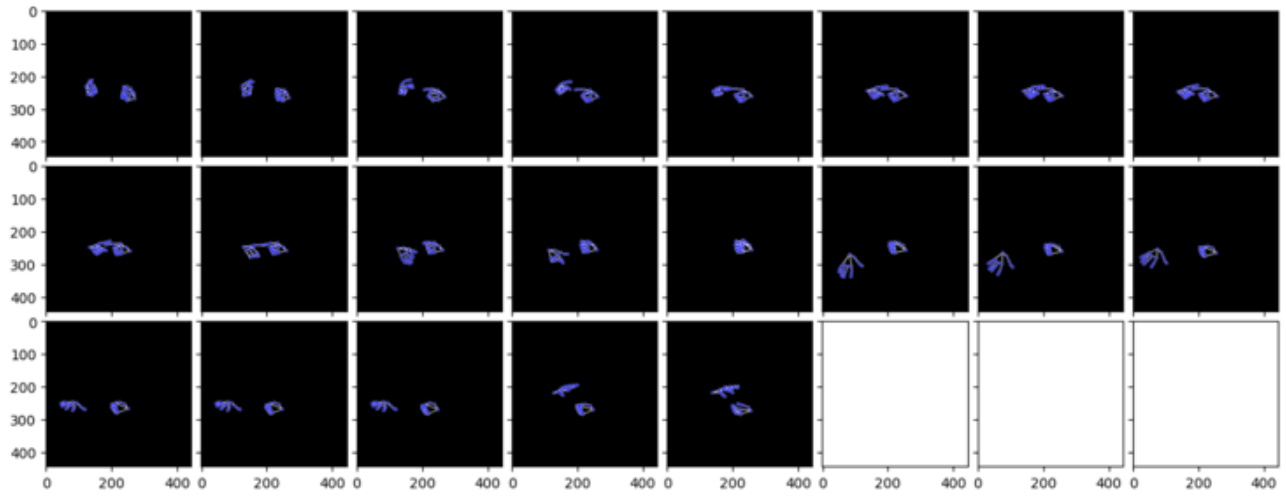

Figure 33: The frame representation of instance of the sign for "family", which does not appear in this sample.

The sign for "family" is a single open hand, with the palm facing downwards, moving in horizontal circles in front of the signer's chest. As we can see in Figure 33, this movement is not present. It does, however, appear that the start of this movement may be shown in the final two frames. This could indicate that for some samples, more than a second elapses between the labelled start time of the sign and the time at which the sign begins.

Although there are some cases like this, the majority of samples in this dataset are correctly labelled, as one would expect from verified data. The cause of the lack of accuracy therefore appears to be surplus data. From analysing the frame representations of this verified data, each sign lasts between 8 and 16 frames. On average, this is equivalent to half a second in real time. This speed is characteristic of professional signers. Despite this, the unpredictable offset between the labelled time and the start of the sign means that the surplus hand location data, in frames before or after the sign is performed, cannot be easily filtered out. It may be possible to align similar frames in each of the samples in an attempt to align the signs, and then filter out the extra data; however, this is not explored in this project.

6.7 6 Words - Unclean, Unverified, 24 Frames

6.7.1 Changes

Based on the findings of the previous iteration, the focus of this iteration was to determine whether increasing the number of frames, and thus the length of each sample, would improve the accuracy of the model on the original word set of 6 words. Increasing the duration of each sample should mean that some samples, which were previously thought to be labelled incorrectly, would now include the correct sign. We also increase the maximum number of samples per sign from 400 to 600.

We expect the accuracy for this iteration to be lower than that of the previous iteration, as the samples used are unverified. If the accuracy is similar, it would indicate that the data available here is almost as clean as the verified dataset. This would be significant, as the unverified dataset contains a much larger vocabulary, and a much larger number of samples for each sign. Of samples initially chosen for verification, Albanie et al. found that 75.8%, with confidence values over 0.8, were labelled correctly[22]. Therefore, we expect approximately 75% of the samples from this dataset to be correctly labelled. This value may vary based on the landmark extraction technique if hands are not present in the first 6 frames, or if a significant number of signs appear more than a second after the labelled timestamp.

As the technique for extracting 24 frames of data allows for missing hands at the start of the clip, there are more samples of each word available than in previous iterations which used this word set. In combination with the extra data in each sample, and the likelihood of more correct labels, we expect an accuracy much higher than that of the first model using this word set (see Section 6.3). The primary comparison, therefore, is with the model developed in Section 6.5, which has an accuracy of 50.8%. In qualitative analysis, we expect a higher percentage of false positives due to the lack of a non-sign class, and a lower percentage of false negatives due to the drastically increased sample size.

6.7.2 Quantitative Results

In order to ensure even further that the model was not overfitting, validation data was used in the training stage for this iteration. This is because the sample size is not as constrained as in previous tests. In each of the 10 folds, approximately 1700 samples are used for training, whilst approximately 200 are used for testing. An initial run using 10% of the training data for validation and 200 epochs revealed that beyond approximately 45 epochs, the model begins to overfit. At this point the accuracy on the validation set is approximately 45%. This graph can be seen in Figure 34.

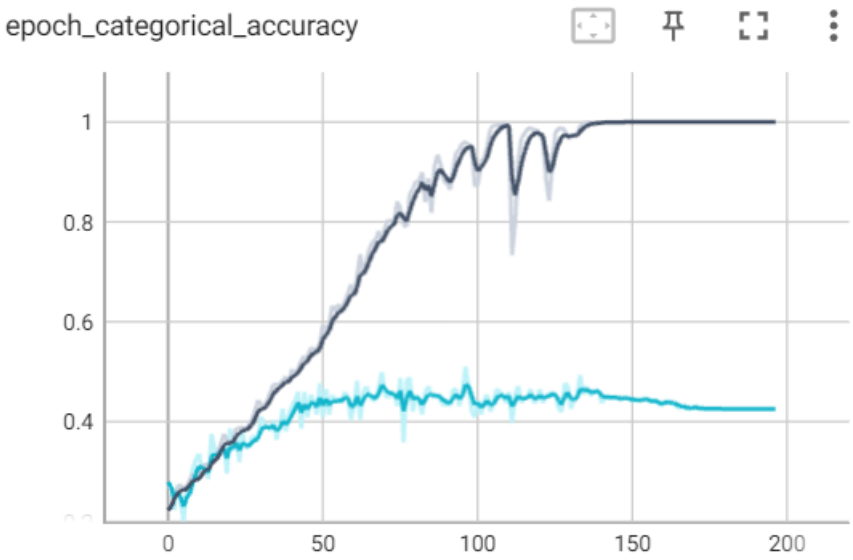

Figure 34: The accuracy of the model for 6 words over 200 epochs, alongside the accuracy on the validation set at each epoch (light blue).

These results tell us that models developed in previous iterations were likely overfitting, although this appears to have little effect on the accuracy recorded - Figure 34 suggests that overfitting on the training data does not affect the model's ability to classify unseen data. This does also suggest, however, that there is an upper limit to the accuracy of this model on test data. One likely cause of this is the noise in the data which has previously been discussed, such as mislabelled signs, and the frames before or after a sign is performed. The latter may be particularly impactful, as all samples are likely to contain similar frames in which hands are either moving between signs or stationary in front of the signer's stomach.

Based on this testing, the 10-fold cross validation for this iteration uses 45 epochs for each fold. The average accuracy achieved on the test data was 45.2%, which correlates with the accuracy seen on the validation data in the previous test. One of the folds, which happened to achieve this exact accuracy, resulted in confusion matrices as seen in Figures 35 and 36.

Figure 35: Confusion matrices for the original 6 words using 24 frames.

		Predicted					
		Hello	You	All-right	Some	Food	Want
Actual	Hello	9	4	4	1	1	2
	You	2	14	5	3	1	10
	All-Right	0	6	12	5	0	4
	Some	0	1	4	18	10	6
	Food	0	4	1	10	18	1
	Want	0	13	1	8	1	18

Figure 36: The detailed confusion matrix for 6 words using 24 frames.

The red highlights in Figure 36 identify the largest sources of confusion. As we can see, the confusion is symmetrical, showing that "some" and "food" are often mistaken for each other, and that "you" and "want" are often confused. The former confusion is expected due to the similarity in movements discussed in Section 6.3. However, there is no similarity been the latter pair of signs, one of which is pointing to the camera whilst the other is the movement of the hand down the chest of the signer and rotating it so that the palm faces the floor. This confusion is likely due to problems with the dataset, as anticipated.

The small sample size used in Section 6.5 makes direct quantitative comparisons difficult, yet this model is 5% less accurate. This suggests that the presence of a non-sign class is beneficial not only to the user experience by reducing the number of false positives, but also in the measurable accuracy of the model. This also indicates that a small but clean set of samples results in a model with greater accuracy than a large set of samples with some noise.

This model also marks a 10% reduction in accuracy over the previous iteration, which used the same approach, albeit on verified data. This supports the theory that approximately 25% of the data is noisy or mislabelled. Due to the large number of samples, manual cleaning of the data is next to impossible.

6.7.3 Qualitative Results

In live testing, this model performs exactly as the results would suggest. Signs are correctly identified approximately half of the time, whilst the other half, they are classified seemingly at random as one of the other signs, even with a high threshold value of 98% confidence. Additionally, there are many false positives which occur when the signer's hands are between

signs or even stationary in front of the signer's stomach. This further suggests the presence of misclassified data within the dataset.

This suspicion is confirmed as the data is analysed once more. Although the frequency of misclassified data is noticeably lower than it was using only 8 frames, there are still some which do not contain any signs at all. For example, the sample shown in Figure 37 is one mislabelled instance of the sign for "hello", which clearly shows no movement of the hands from in front of the signer's stomach. It is likely that there are other similar samples, which explains why the model sometimes gives false positives when the signer is stood like this. The same is true for other false positives in which no sign is being performed.

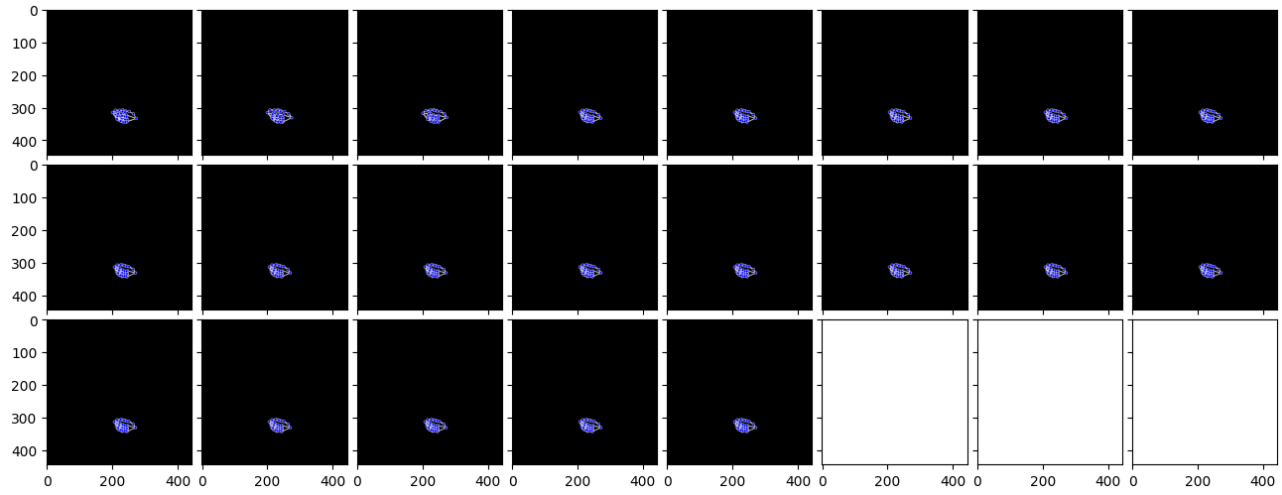

Figure 37: 24 frame images representing one mislabelled sample of the sign for "hello".

6.8 6 Words - Verified, 24 Frames, With Pose Data

6.8.1 Changes

This final iteration aims to give the model even more data to work with by using both pose and hand landmark detection. This means that the data now has 258 features. The reasoning behind this decision is that the model will learn to use the movement of the hand relative to the body in order to classify signs, rather than the movement relative to its position in the frame. This would be particularly beneficial for signs such as "food" and "look", which use the signer's face, and should enable the classification of signs when the signer is not central in the frame.

The verified dataset is used in this iteration due the lack of mislabelled data, and the relatively good performance of the previous model which uses it. The word set used for this test can be seen in Table 2.

The process of including pose data is very simple. We apply the pose landmark detection model to each frame of a video and store the results alongside the hand landmarks detected for

that frame. We can then store and read this data exactly as before. The neural network's input layer was changed from a size of 126 to 258 to account for the increased number of features, and the analysis notebook script was modified so that each frame displays both the pose and the hands.

The sample size is reduced to 200, as we have seen no evidence that larger sample sizes result in increased performance of the model. The number of epochs is also adjusted as before to reduce overfitting.

6.8.2 Quantitative Results

As in the previous iteration, validation data is used in the initial training of the model, along with a large number of epochs, in order to see at what point the model begins to overfit. The results of this can be seen in Figure 38.

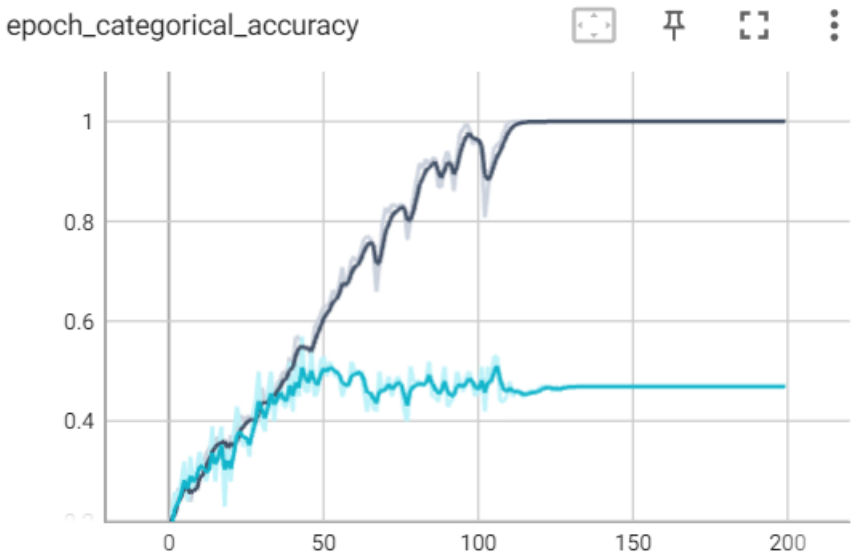

Figure 38: The accuracy of the model on 6 words from the verified dataset with pose and hand landmark information. Light blue depicts accuracy on the validation set.

As we can see in this graph, the accuracy of the model on the validation set peaks at approximately 55 epochs. We also note that it does not significantly decrease as the model begins to overfit to the training data. The model was therefore tested with 10-fold cross validation using 55 epochs. The average accuracy was 44.4%, a significant drop from the 55.6% accuracy of the previous model which used only hand landmarks. The cause of this is unclear at this point, although it is likely a result of the increase in data which is not relevant to classification. For example, the pose information will be almost consistent across all samples, with the most variation in the arms, which account for only 12 of the 33 landmarks that make up the pose. Additionally, the signer is only visible from the waist up, which means that all landmarks below the hips are likely to have visibility values of 0. This large amount of redundant data may cause confusion between classes, which will all have similar data for these

landmarks. This could be remedied by using dimensionality reduction techniques, although this is not explored in this project.

The confusion matrices in Figures 39 and 40 demonstrate the confusion between signs. When comparing to the confusion matrix shown in Figure 32, we can see that while the primary sources of confusion remain the same, there is much additional confusion between all other signs. This supports the theory that the inclusion of pose data results in the model using data which is consistent across many samples from different classes, which in turn causes a lower overall accuracy.

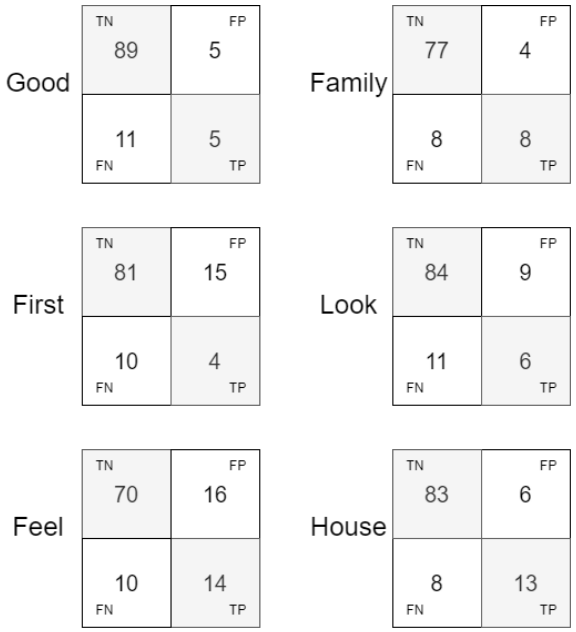

Figure 39: Confusion matrices for 6 words from the verified dataset using pose and hand data.

		Predicted					
		Good	First	Feel	Family	Look	House
Actual	Good	5	0	4	2	2	3
	First	0	4	0	1	9	0
	Feel	3	4	6	3	0	1
	Family	1	3	2	6	4	2
	Look	1	5	2	2	14	0
	House	0	3	1	3	1	13

Figure 40: The detailed confusion matrix for 6 words from the verified dataset using pose and hand data.

6.8.3 Qualitative Results

Live testing was not performed in this iteration, as the quantitative results indicate significant levels of confusion, and lower accuracy than that obtained by previous iterations. The use of pose data should therefore not be continued unless the data source used changes, and more of the signer's body is present in the video.

6.9 Results Summary

6.9.1 Quantitative Results

Throughout development, the accuracy of the model has been dictated by one primary factor, which is the quality of the data. The neural network design has seen very few changes, whilst the majority of iterations have focused on employing different pre-processing steps, using different

data, or manual cleaning of the dataset. We also note that while early models are overfit to the training data, evidence suggests that this has little to no effect on the accuracy of the model on test data.

The results obtained are unfortunately not representative of the capability of the neural network design, but rather serve to highlight the importance of reliable data. This is further emphasised by the qualitative testing performed, as improving the quality of the data, whilst having little effect on the achieved accuracy, allowed for a model which was far more reliable and useful in live detection due to the presence of a non-sign class (see Section 6.5). We therefore suggest that the accuracy of each model is not considered independently, but rather that they are compared to each other with regards to the changes made between each iteration. The accuracy of each iteration can be seen in Table 3. Note that this omits the third iteration, which was not representative of the model nor the available data due to the unbalanced nature of the dataset used.

Model	**Accuracy**	**Qualitative Comments**
4 Words (Section 6.2)	52%	Unremarkable
6 Words (Section 6.3)	36%	Many false positives
6 Words - Clean (Section 6.5)	50.8%	Improved recognition of signs, false positives filtered by non-sign class
6 Words - Verified, 24 Frames (Section 6.6)	55.7%	Confusion in similar signs, very good for distinct signs
6 Words - Unverified, 24 Frames (Section 6.7)	45.2%	Many false positives from non-signs
6 Words, - Verified, 24 Frames, With Pose Data (Section 6.8)	44.4%	Untested - likely high confusion

Table 3: The accuracy of the major iterations of the model, and comments on the qualitative performance in live sign detection.

These results are much lower than we initially anticipated based on the performance of the model proposed by Samaan et al.[15]. This may be due to the technical implementation of the system, which is omitted in their paper. It is, however, more likely caused by the structure of the data. A custom dataset allowed them to have complete control over variables such as sign length, sign speed, distance from the camera, hand occlusion, and lighting conditions. This allowed them to minimise noise in the dataset, both from useless frames in the video either side of the sign itself, and from mislabelled samples. The closest comparison to this is the verified dataset, in which each sample is known to be labelled correctly, although some of the signs extend beyond the 24 frames collected in the sample, as discussed in Section 6.6. Including extra information in the form of pose landmarks resulted in a significant decrease in accuracy, as seen in Section 6.8. We therefore conclude that we are unable to reproduce or come close to the results obtained by Samaan et al.[15].

There are several techniques which were not applied in this project which could help to achieve a higher accuracy. These are discussed in Section 7.3.

6.9.2 Qualitative Results

The iterative process allowed for repeated qualitative testing, and thus comparison of the similarities and differences between each model. The limitations of the dataset present themselves very clearly in live testing. For example, the position of the signer in the frame and their distance from the camera is very important. As the dataset contains only professional signers on BBC televised broadcasts, these variables are strictly controlled. The model performs most reliably in live testing when these conditions are mimicked and sees particular loss in performance when the signer is much closer or further away from the camera. This is caused by the differences in the numerical coordinate values of landmarks, which become much greater or smaller respectively.

It is also evident from live testing that professional signers are much stricter with the movement of their non-dominant hand than non-professional signers. While a daily user of BSL may drop their non-dominant hand to the side if it is not being used, all professional signers in the dataset return any hands not in use to the front of their stomach. This contributes to the confusion of many models, as almost all samples will contain some frames in which one or both hands are in front of the signer's stomach. This position may not be natural to all users of sign language and means that the model sees improved performance when the signer is strict with their hand positions.

We also note that a very high confidence threshold was used for all live testing, of either 95% or 98%. Despite this, all iterations of the model experienced a high number of false positives, highlighting the issues with the data. The best results in live testing were achieved by the manually cleaned data. This is due to the introduction of a non-sign class, which was not output to the user. When non-signs were detected, such as the movement of the hand between signs, or when hands were stationary, this model did not give a false positive for one of the actual signs, allowing for a significantly improved user experience. The model was not perfect, as the vastly reduced sample size meant that true positive classification was notably limited, with signs often not being recognised.

We therefore conclude that whilst the verified dataset resulted in the highest accuracy on individual signs, the manually cleaned dataset resulted in a better user experience for continuous signing.

7 Conclusions

In this section, we analyse the execution and results of the project against the previously-established objectives and plans. We also consider situations which should have been handled differently, or in which mistakes were made. We additionally discuss future work, which includes both extensions to the project which were not planned, and extensions based on objectives which were not completed.

When evaluating the performance of the model, assume references are to that developed in Section 6.6 unless specified otherwise. This iteration achieved the best performance in quantitative testing and was among the best performers in qualitative analysis.

7.1 Objectives

Many of the objectives defined in Section 3 are either incomplete or only partially complete. There are several reasons for this, ranging from poor project management to unanticipated problems. In this section we summarise these objectives, determine whether or not they were achieved, and analyse their importance in the final result of the project.

7.1.1 Must-Have Objectives

All objectives which were considered must-haves were completed entirely.

M1 The live detection program which was used for qualitative analysis can also access a recorded video if required by simply changing the video capture source.

M2 As the detection of hands in each frame is handled by the MediaPipe library[13], the performance is extremely reliable. Although problems arise when hands are occluded or motion blur obscures them, these issues have been addressed by using the hand's last known location. Though this objective is complete, the accuracy could be improved by estimating hand locations using their trajectory and movement through previous frames.

M3 The Recurrent Neural Network implemented is capable of classifying signs in a video feed with up to 55.7% accuracy and outputs them to the user as text. Logic is also implemented to ensure that each sign predicted is output exactly once in succession so that the same movement is not classified multiple times in different frames.

M4 The program is capable of classifying signs extremely quickly. The application of the model to data in a live feed takes approximately 20ms, which means that the user receives a classification almost instantly.

As all the must-have objectives were completed, the software developed meets the minimum required expectations. Despite this, these objectives do not encompass the accuracy or perfor-

mance of the system, and so do not inherently ensure that the proposed solution fills the gap in recent research.

7.1.2 Should-Have Objectives

The objectives which were considered should-haves aimed to ensure that the system created was not only functional, but also effective and simple to use. Many of these objectives were not met, or only met partially due to the challenges and unexpected obstacles encountered during development.

S1 This objective was not met on both fronts. The proposed solution is only able to identify 6 signs, and with an accuracy of only 55.7%. This is significantly fewer than the 20 signs initially identified, and the accuracy is also significantly lower. These numbers were established as targets based on background reading in which similar approaches were able to achieve very high accuracy on similar problems[15, 10, 14]. This objective was not met due to issues with the quality of the data used.

S2 This objective was partially met. The use of the MediaPipe library means that the lighting and camera quality has little effect on the classification of a sign. In the event that these conditions result in MediaPipe being unable to identify hands in the image, previous locations are used. The effects are therefore minimised, and these conditions are not significant in sign classification.

The distance of the signer from the camera, and their position in frame, are two variables which were not properly addressed in this project. The data passed to the model are coordinates relative to the frame, which means that signer position and distance from the camera has a significant impact on the classification of signs, and this objective is not complete.

S3 As the neural network used is a Recurrent Neural Network, it is capable of using multiple pieces of data to generate one classification. The proposed solution can generate an output for any number of frames, and so this objective has been met.

S4 This objective has been partially met. The default camera source is 0, and the user is able to modify this in the script if they have multiple camera devices. OpenCV[37] does not currently contain a simple way of identifying camera sources, and so the user cannot select the source once the program is run.

S5 Due to the use of the MediaPipe library, this objective has been entirely met. The skin colour of the signer has little to no impact on the extraction of hand landmarks, as evidenced by the various signers in the source dataset from which MediaPipe was able to gather data.

S6 This objective was not met at all. Due to the unanticipated problems with the dataset, alongside time management issues, development of the neural network extended much longer than expected, and thus the component of the system responsible for the formation short sentences using Natural Language Processing was cut from the scope of the project.

The failure to meet many of these objectives, particularly S1, means that the proposed solution does not fill the identified gap in research, which was a model capable of accurate recognition of dynamic signs involving two hands. Despite this, the use of MediaPipe to achieve objective S5, combined with the promising albeit low 55.7% accuracy achieved by the model suggests that a signer-independent sign recognition system is feasible with sufficient high-quality data, time, and iterations.

7.1.3 Could-Have Objectives

Due to the aforementioned problems encountered, the could-have objectives have not been considered at all in the development process.

C1 By default, MediaPipe recognises 2 hands in an image and tracks them through the video. In a perfect situation, other people in the frame would have no effect on the system. This often does not happen though; if a hand is undetected due to motion blur or occlusion, MediaPipe searches for new hands to track. This means that other people in the frame may have their hands tracked instead. Therefore, this objective is partially met.

C2 As the model developed is not capable of recognising with high accuracy even 6 signs, it has not been extended to a larger vocabulary.

As it was not anticipated that these objectives would be completed, the fact that they have not been does not have much consideration in the success of the project. The extension to a larger vocabulary, in particular, is a focus of Section 7.3.

7.2 Project Management Reflection

Section 4.1 outlined the development methodology with which the project would be executed. The iterative methodology was executed effectively, with regular iterations, each of which tackled a specific task and contained detailed evaluation of the progress and changes made. The application was very agile in practice, as many iterations focused on solving problems discovered in prior iterations, such as the cleaning of the data, or switching to verified data. This aspect of project management was successful, as there were no points at which the next step was unclear.

While the execution of the development methodology was effective, the time management techniques were not so well executed. Progress was inconsistent, particularly around periods when other coursework was due, which meant that the project often fell behind the schedule

defined in Figure 42. The rate of work then had to be increased in order to catch up. Furthermore, progress frequently fell behind schedule due to unanticipated problems, primarily those with the dataset. These problems should have been anticipated much sooner, as further analysis of the dataset prior to the beginning of development would have revealed many problems. If this had happened, a custom dataset could have been created, which would have been free of the problems present in the BOBSL Dataset[22]. When combined with the inconsistent working hours, these problems meant that development took much longer than anticipated, and together constitute the reasoning behind the cutting of the sentence formation feature.

The tools used to help manage the project were extremely effective and useful. GitHub allowed not only for backup, but also for working across three different machines, and OneDrive helped to ensure that the most recent version was always available no matter the device, even if it had not yet been pushed to GitHub.

Overall, the project was well managed; however, time constraints and poor planning led to some rushed decisions and mistakes. All of these have been documented throughout the report and served as opportunities for learning. By the end of the project, progress was consistent, although slow.

7.3 Future Work

In this section, we describe several opportunities for both improvement and extension to the project. Some of these were initially planned inclusions that were cut due to time constraints.

7.3.1 Improved Accuracy

The primary goal of future work should be to improve the accuracy of the model on test data. There may be many methods of achieving this, such as using a different neural network model entirely as some others have done[10, 11], or by using a different dataset[15]. Though we found that manual cleaning of the data in the BOBSL Dataset resulted in the largest improvement in accuracy, this is not feasible for a larger number of samples. Therefore, we propose that the first step towards improving the accuracy achieved by the model is to use a custom dataset, which is free of noise.

Another method of improving the accuracy may be to use MediaPipe's world landmarks rather than the normal landmarks. These represent the same locations on the hand, but the coordinate system is measured in meters, with the origin at the approximate centre of the hand. This could allow the model to learn hand positions in more detail, although the data would no longer reflect the relationship between the positions of the hands, in regards to the signer or each other. Therefore, a better solution may be to also apply pose detection to detect the signer's approximate position and calculate each hand's landmark's positions relative to a point on the signer, such as their face or the centre of their waist. This would ensure that the information not only reflects the orientation and pose of each hand, but also where they are in

relation to the signer and each other. Using this approach, the system could also be invariant of signer location in the frame or distance from the camera if these distances were somehow normalised.

Dimensionality reduction techniques should also be applied to the data in order to reduce the impact of the irrelevant frames and coordinates. With only hand coordinates, each frame contains 126 features, which means that over 24 frames, there are 3024 features being taken into account. As there are several signs which use only one hand, if the non-dominant hand is stationary throughout the sign, then half of the features are effectively meaningless and provide very little information required for the classification of the sign. Therefore, applying dimensionality reduction techniques such as Principal Component Analysis[41] in order to better represent the data could significantly improve accuracy. Additionally, more pre-processing steps should be employed to reduce the amount of useless data passed to the model by duplicate frames, or frames in which there is little movement. This would ensure that extended periods of little motion are not used to classify the sign.

A final suggestion for improving accuracy would be to train separate models for single handed and two-handed signs. In this way, we could vastly reduce the confusion which is a result of all one-handed signs also having a stationary non-dominant hand present in the image. One model may have an input layer with 63 neurons, capable of tracking movement of one hand, whilst another would have 126, capable of tracking the movement of both hands. An additional machine learning model or complex logical algorithm could be used to determine which of these models the data is passed to. This composite model could result in an increased accuracy due to the clear distinction between one-handed and two-handed signs.

7.3.2 More Words

An obvious extension to this project is to increase the number of words on which it is trained and tested, though this should not be attempted until a model capable of recognising the current set of words with high accuracy is developed, as described in the previous section. By including more words, we could explore how the model adapts to a larger variety of signs, particularly those which use the dominant hand and the non-dominant arm for example, which were not tested in this project. Additional words should also include more which use both of the signer's hands in order to determine how often signs which use both hands are confused.

A natural long-term extension of the word set using the BOBSL Dataset would be to the 91 words identified by Buehler[12] on which an accuracy of 39.5% was achieved. Buehler's work uses the same dataset, and only the first and final frame of a sign for classification. The use of technology such as MediaPipe to reduce the impact of varying signer skin colour and mannerisms, along with techniques to improve the accuracy of the model as suggested in the previous section, should allow development of a model capable of matching or surpassing the accuracy achieved by Buehler in a signer-specific setting, which was 57.9%.

The word set could also be extended to include static signs, such as the alphabet. This would allow for more direct comparison to many other existing works[9, 10, 14]. It should, however, be noted that the proposed model may not be suitable for this task. Single letters are often signed extremely fast due to their stationary nature, and so may only constitute a fraction of a second. 24 frames would therefore be too long and would possibly also include part or whole of the following sign if a word is being fingerspelled.

7.3.3 Sentence Recognition

Another area in which this project could be further developed is the formation of sentences from sequences of signs. As English and British Sign Language have different grammar systems[42], this task would involve the use of complex natural language processing (NLP) systems. As BSL lacks articles, such as "a" and "the", as well as different forms of verbs or nouns to convey tense and plurality, a system must be able to use the context of a sentence to insert the required words. It must then restructure the sentence according to English grammar rules and use some form of reverse tokenization to convert the individual words into a sentence. As sentences may be of variable length, a system must also be able to dynamically recognise the start and end of sentences. This task is therefore extremely complex, and likely requires the use of a bi-directional recurrent neural network in order to ensure that all context clues are used in determining which articles, verb and noun forms are inserted into each sentence.

7.3.4 Other Signed Languages

Due to the nature of sign language, which primarily uses the hands, the computational problem of individual sign interpretation can be easily compared to gesture recognition[43]. This is a common trait of signed languages and is not language specific. Therefore, the research and solution proposed in this project could, with relative ease, be extended to other languages such as ASL. As discussed in Section 1.2, there is more to the translation of sign language than just hand gestures, as facial expressions and the mouthing of words also impact the interpretation of a sign. This is also a feature of all signed languages, and therefore the limitations of the model presented in this work are also applicable to other languages.

For the purposes of research and developing an effective solution for sign recognition, it would likely be beneficial to develop a system trained on a sign language other than BSL, due to the lack of high-quality datasets. Therefore, other sign languages, many of which may contain similar features to BSL, such as a two-handed manual alphabet, may provide sufficient data to create a more accurate sign recognition system, which could then be adapted for the task of recognising British Sign Language. Alternatively, the formation of a word-level BSL dataset containing videos of many signers, with manual labelling of data, would nullify many of the challenges faced in this project, and thus provide a better foundation for the creation of a sign language translation system.

7.4 Conclusion

The system developed throughout the course of this project, while providing results which fail to match the accuracy achieved by others, highlights the importance of using high-quality data which is suited to the task. The process of iterative development revealed many problems with the dataset chosen, which were not obvious during initial research. This project therefore serves to explore the impact of data and various pre-processing steps on a machine learning model which is capable of a high classification accuracy[15].

This project is also evidence that signer-invariant translation systems are feasible, as the use of MediaPipe allows for accurate hand, pose, and facial detection in any image or video. We have shown that this data can then be used for sign classification, although more pre-processing steps are necessary in order to improve classification accuracy. We have also shown, using qualitative analysis of multiple machine learning models, that individual sign recognition is not sufficient for an acceptable user experience. In particular, the system developed in Section 6.5, which makes use of a non-sign class, provided the best qualitative results, as it was able to recognise gaps between words or sentences, and thus reduce the number of false positives for signs.

Throughout the course of the project, many lessons have been learnt, although there were also some decisions which were proven to be incorrect. The main issue is that the dataset used throughout development was not suitable for the task and should have been replaced with a custom dataset as early as possible. The decision to continue using the BOBSL dataset meant that far more development time was spent tackling issues with the data than initially intended, which in turn meant that the implementation of the neural network is very simple and did not receive many adjustments.

In conclusion, the system developed is capable of recognising 6 signs with an accuracy of 55.7%. This is a far lower number of words and accuracy than initially planned. Speculative suggestions of methods for improving both accuracy and the size of the word set have been laid out in Section 7.3, and other works released during the course of this project suggest that with higher quality data, an accuracy as high as 100% is achievable[15].

References

[1] "Work to be done on census figures for bsl," 2022. [Online]. Available: https://bda.org.uk/bsl-census-figures-2022/

[2] M. Ravenna, "10 facts about british sign language," Mar 2018. [Online]. Available: https://www.k-international.com/blog/british-sign-language-bsl-interpreters/

[3] SignHealth, "What is the difference between deaf and deaf?" Mar 2020. [Online]. Available: https://signhealth.org.uk/resources/learn-about-deafness/deaf-or-deaf/#:~:text=The%20word%20deaf%20is%20used,started%20to%20learn%20to%20talk.

[4] S. U. Team, "What's the difference between asl & bsl?" Jul 2021. [Online]. Available: https://www.skill-up.org/whats-the-difference-between-asl-bsl/#:~:text=ASL%20and%20BSL%20are%20Different%20Languages&text=But%20the%20most%20significant%20difference,of%20the%20alphabet%20and%20numbers.

[5] R. Sutton-Spence and B. Woll, *Sign Order*. Cambridge University Press, 2002, p. 50.

[6] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in british sign language," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2009, pp. 50–57.

[7] J. Olszewska and M. Quinn, "British sign language recognition in the wild based on multi-class svm," 09 2019, pp. 81–86.

[8] R. Rastgoo, K. Kiani, and S. Escalera, "Sign language recognition: A deep survey," *Expert Systems with Applications*, vol. 164, p. 113794, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030614X

[9] B. Sonare, A. Padgal, Y. Gaikwad, and A. Patil, "Video-based sign language translation system using machine learning," in *2021 2nd International Conference for Emerging Technology (INCET)*, 2021, pp. 1–4.

[10] K. Kumar, "Deaf-bsl: Deep learning framework for british sign language recognition," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 21, no. 5, aug 2022. [Online]. Available: https://doi.org/10.1145/3513004

[11] R. Elakkiya, P. Vijayakumar, and N. Kumar, "An optimized generative adversarial network based continuous sign language classification," *Expert Systems with Applications*, vol. 182, p. 115276, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421007077

[12] P. Buehler, "Automatic learning of british sign language from signed tv broadcasts," Ph.D. dissertation, Oxford University, 2010.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for perceiving and processing reality," in *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019. [Online]. Available: https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf

[14] A. Ray, S. Syed, S. Poornima, and M. Pushpalatha, "Sign language recognition using deep learning," *Journal of Pharmaceutical Negative Results*, p. 421–428, Sep. 2022. [Online]. Available: https://www.pnrjournal.com/index.php/home/article/view/518

[15] G. H. Samaan, A. R. Wadie, A. K. Attia, A. M. Asaad, A. E. Kamel, S. O. Slim, M. S. Abdallah, and Y.-I. Cho, "Mediapipe's landmarks with rnn for dynamic sign language recognition," *Electronics*, vol. 11, no. 19, p. 3228, Oct 2022. [Online]. Available: http://dx.doi.org/10.3390/electronics11193228

[16] B. Walters, "Project specification," 2022.

[17] V. R. Basil and A. J. Turner, "Iterative enhancement: A practical technique for software development," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 4, pp. 390–396, 1975.

[18] github, "Github," 2020. [Online]. Available: https://github.com/

[19] B. Walters, "Progress report," 2022.

[20] M. Jay, "American sign language: Start asl," Feb 2021. [Online]. Available: https://www.startasl.com/american-sign-language/

[21] A. Schembri, J. Fenlon, R. Rentelis, and K. Cormier, "British sign language corpus project: A corpus of digital video data and annotations of british sign language," 2008. [Online]. Available: https://www.bslcorpusproject.org/

[22] S. Albanie, G. Varol, L. Momeni, H. Bull, T. Afouras, H. Chowdhury, N. Fox, B. Woll, R. Cooper, A. McParland, and A. Zisserman, "BOBSL: BBC-Oxford British Sign Language Dataset," 2021.

[23] C. Neidle and C. Vogler, "A new web interface to facilitate access to corpora: Development of the asllrp data access interface," in *Proceedings of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon*, 2012.

[24] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali, "The asl lexicon video dataset," *CVPR 2008 Workshop on Human Communicative Behaviour Analysis*, vol. 4, 2008.

[25] D. Li, C. Rodriguez, X. Yu, and H. 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*, 2020, pp. 1459–1469.

[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[27] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," *CoRR*, vol. abs/1711.11248, 2017. [Online]. Available: http://arxiv.org/abs/1711.11248

[28] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[29] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[30] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Advances in psychology*. Elsevier, 1997, vol. 121, pp. 471–495.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] TensorFlow, "Recurrent neural networks (rnn) with keras & tensorflow core," Aug 2022. [Online]. Available: https://www.tensorflow.org/guide/keras/rnn

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[34] L. Lu, "Dying ReLU and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, jun 2020. [Online]. Available: https://doi.org/10.4208%2Fcicp.oa-2020-0165

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[36] K. Kroening, "ffmpeg-python: Python bindings for ffmpeg," 2017. [Online]. Available: https://kkroening.github.io/ffmpeg-python/

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

[38] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[39] N. Renotte, "A practical implementation of sign language estimation using an lstm nn built on tf keras." 2021. [Online]. Available: https://github.com/nicknochnack/ActionDetectionforSignLanguage

[40] 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*, vol. 12, pp. 2825–2830, 2011.

[41] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.101

[42] R. Sutton-Spence and B. Woll, *Linguistics and Sign Linguistics*. Cambridge University Press, 2002, p. 10–10.

[43] M. Turk and V. Athitsos, *Gesture Recognition*. Cham: Springer International Publishing, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1007/978-3-030-03243-2_376-1

A Figures

Figure 41: Some example signs which are referred to in this paper. Generated using BSL Vocabulary Sheet Creator (https://www.british-sign.co.uk/bsl-vocabulary-sheet-creator/).

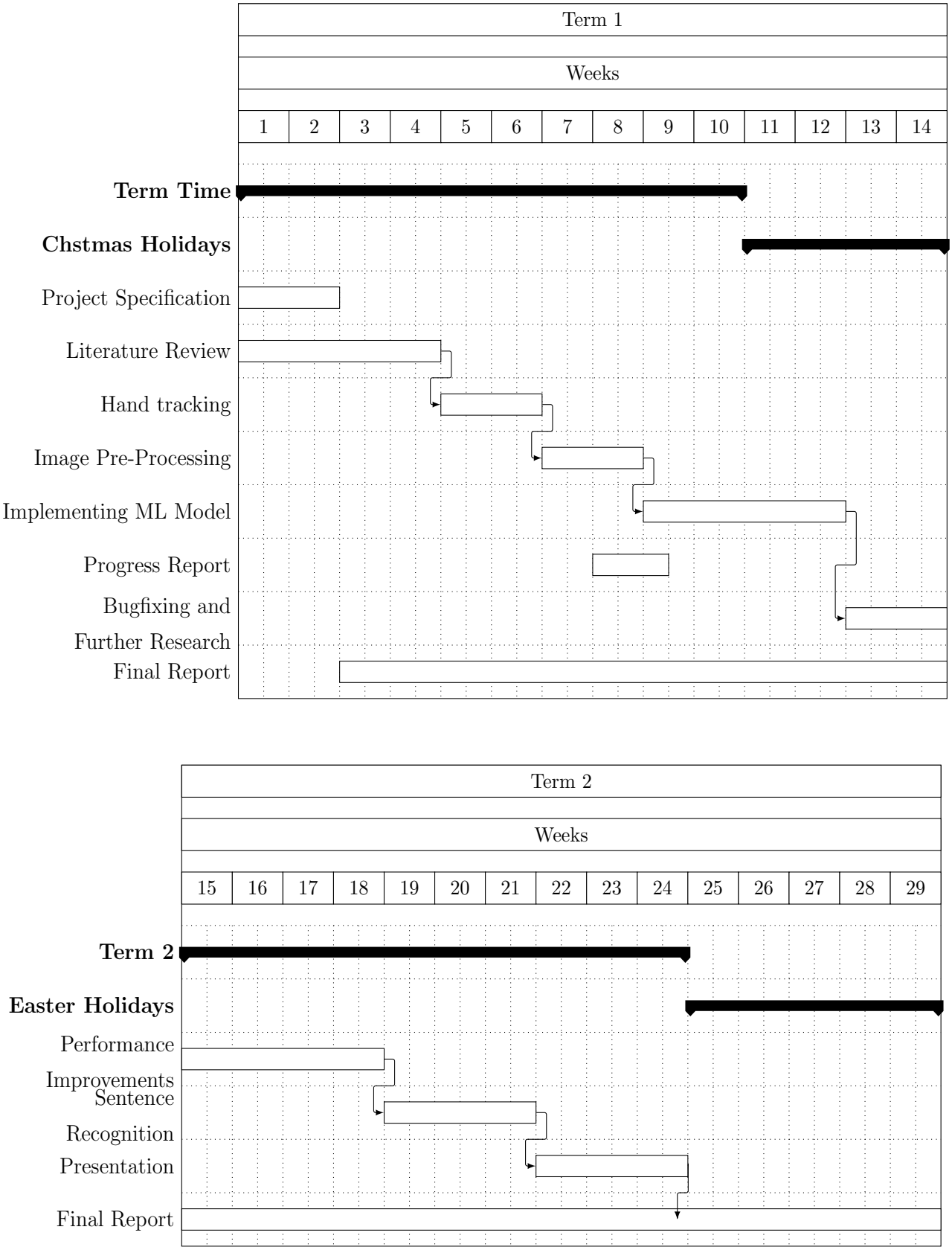

Figure 42: Gantt Chart representing the planned project timeline.