Machine Learning: Translating sign language in real time.

Oliver Storey-Young

supervisor: Michael Daley

MSc Advanced Computer Science

School of Computer Science and Informatics

Cardiff University, Wales

I Abstract

As the ability to access technology continues to grow, we have seen a large increase in the demand for video calls. Allowing people to connect at a more personal level than a phone call, video calls allow family, friends, and employers to communicate worldwide, face to face, over the internet. With the pandemic came an increased demand for this and similar services, as people were unable to visit each other in person.

Statistics collected suggest that in 2011 around 151,000 UK residents used British Sign Language(BSL)[1]. This project aims to create a software product that demonstrates the ability for accessibility in this or a similar scenario for those BSL users. The product will allow for users to receive an accurate and fast translation of their signs, which could be forwarded on in a live conversation.

The outcome of the project was an application that makes use of a webcam and Google's MediaPipe[2]. The product uses machine learning to interpret a sequence of actions performed by a user based of a set of possible signs it has been trained on. A pair application to go with this can be used to capture more training data if needed. The application performs well and could be adapted to make use of other localisation's such as American Sign language(ASL)[3].

II Acknowledgements

I would like to give a big thank you to my Mother who has supported me through the creation and completion of this project and dissertation. I would also like thank my supervisor, Mike Daley, who has helped guide me through the twelve weeks!

This MSc dissertation marks my final hand-in at Cardiff University. I came to Cardiff five years ago, not knowing anyone here, or anything about the city. I now leave with a wealth of knowledge, memories, friendships and an exciting new career to pursue. I would like to give a final big thanks to all of those people who have helped make me who I am today.

Contents

I Abstract 2
II Acknowledgements 3
III Introduction 9
III.I Project Overview 9
III.II Project Aims 10
III.II.I First aim 10
III.II.II Second aim 11
III.II.III Third aim 11
III.III Project Scope 11
IV Background 13
IV.I Sign Language 13
IV.II Capturing Data 14
IV.III Machine learning 16
IV.IV Comparison to other products 17
V Approach 19
V.I Research 19
V.I.I Image capture 19
V.I.II Image detection 20
V.I.III sequence prediction 21
V.II Foundations 21
V.II.I Setting up the environment 21
V.II.II Creating the User Interface 22

V.III Functionality 22
V.III.I Building the model 22
V.III.II Predicting the sign 23
VI Specification And Design 24
VI.I Data Collection 24
VI.I.I User Perspective 24
VI.I.II System behaviour 25
VI.I.III Architecture 27
VI.I.IV Constraints 27
VI.II Data Storage 28
VI.II.I User Perspective 28
VI.II.II System behaviour 29
VI.II.III Architecture 29
VI.II.IV Constraints 30
VI.III Building the model 30
VI.III.I User Perspective 30
VI.III.II System behaviour 31
VI.III.III Architecture 32
VI.III.IV Constraints 32
VI.IV Translating signs 33
VI.IV.I User Perspective 33
VI.IV.II System behaviour 34
VI.IV.III Architecture 35
VI.IV.IV Constraints 35
VII Implementation 37
VII.I Collecting data 37
VII.I.I Interface 37
VII.I.II Capturing key points 38
VII.I.III Output 39
VII.II Building the model 40

VII.II.I Importing the data 40
VII.II.II The Model 41
VII.II.III Evaluative Techniques 42
VII.III Translation 42
VIII Results and Evaluation 44
VIII.I Evaluation by User testing 44
VIII.II Evaluation by Test cases 46
VIII.III Test cases 47
IX Future Work 53
IX.I Code Work 53
IX.I.I Code-Rewrite 53
IX.I.II Error Handling 53
IX.I.III User Interface Redesign 54
IX.I.IV API 54
IX.II New Features 55
IX.II.I Mobile 55
IX.II.II Exports 55
IX.II.III Fluent Signing 55
IX.III Community 56
IX.III.I Switching Data 56
IX.III.II Crowd-sourcing Community 56
X Conslusion 58
XI Reflection On Learning 60

List of Tables

1 Holistic landmark totals 15
2 Results of user testing 45
3 Test Case 1 47
4 Test Case 2 48
5 Test Case 3 48
6 Test Case 4 49
7 Test Case 5 50
8 Test Case 6 50
9 Test Case 7 51
10 Test Case 8 51
11 Test Case 9 52

List of Figures

1 Examples of MediaPipe Holistic being used. 15
2 The general schema of a RNN unit versus a LSTM one (adapted from Olah 2015)[22] 17
3 Example of bounding boxes displayed over hands[42]. 20
4 Initial Design of data collection tool. 25
5 Implemented User interface of data collection tool with MediaPipe overlay. 26
6 File structure generated and used by the data collection tool. . 28
7 Confusion Matrix generated by the model.ipynb. 31
8 Implementation of user interface for translation application. . 34
9 Code implementation of the data collection layout. 37
10 Code implementation of updating the Image element with the next frame. 37
11 Code implementation for drawing landmarks over camera feed. 38
12 Code implementation of converting Holistic data to desired format. 39
13 Code implementation to sort based on the first item in the list, and then append all data but that value. 40
14 Implementation of the machine learning model. 42
15 Code Implementation showing the sequence of adding frames to the queue, making and displaying a prediction. 43

III Introduction

III.I Project Overview

All over the world people use different languages to communicate. Applications exist to help us translate our voice into text and then into other languages. For a small portion of the world whose primary form of communication is localised sign language, they're often left with a lack of accessibility features in technology. While applications and tools do ultimately exist to bridge impairments and accessibility requirements for different reasons, this project will focus on translating BSL into text.

The applications of this project are very broad and sign language can come in many different forms. This project will be built and modelled around BSL, but should be ultimately interchangeable with different data sets to allow for different sign language localisation's. This product could be used in a variety of different ways such as translating video calls, so BSL users can talk to people who don't understand sign language, it could act as a tool for practicing and learning BSL and could even just be used to save a fluent BSL user from having to type out long sentences.

This dissertation will explore the techniques that can be used to meet the projects aims. It will explore the ideas, background research and logic behind the decisions of the final product. I will compare my solution to my aims and objectives while comparing it to other possible implementations. It will also discuss where my application stands in terms of next steps, future work and reflect on the project as a whole.

From this project I would like to increase my knowledge in both Python and machine learning. This project will use machine learning to reinforce connections between the frames of videos in a BSL sign. This will be a step-up challenge from previous projects I have undertaken as it involves sequences of images and key points rather than machine learning on static data-sets which I have previously done. I also aim to learn some BSL as I will

have to use it when collecting the data myself. Getting an understanding of BSL basics will help me understand the needs of the people who this project aims to help.

III.II Project Aims

The project has been divided up into 3 main aims. These aims make up the core functionality of the program which being met, would warrant a usable and relevant product. The three main aims may change slightly over the course of the project but as I will be using a take on the Agile methodology[4] this can be accounted for and handled accordingly at the time. The three aims briefly are: Translation, flexibility, and speed. Firstly needing an accurate and usable translation will ensure the user is actually gaining something through using our application. Being able to take the translation from the application means it can be sent or used elsewhere. flexibility will mean that our user can add more signs if they decide to. The user could also import data sets and models from other users. Lastly speed, the application is going to be processing predictions and taking a feed directly from the camera, the application needs to handle this without issues else the users experience will be hindered.

III.II.I First aim

When the user performs a sign, a text translation is provided.

- The application clearly displays the output translation to the user.
- The user can copy or export the translation from the application.
- The application makes an attempt to recognise "no sign" or false signs.

III.II.II Second aim

The application is easily adaptable to incorporate more signs and gestures

- The application should be extensible and be able to detect action sequences other than BSL if needed.
- The user should be able to use different data sets.
- The user should be able to record more data for themselves.

III.II.III Third aim

The application runs fast and smoothly

- The video feed of the user is not slowed down by the prediction processing
- The application is able to present a prediction to the user before they move to their next sign (keeping it real time).
- The application Doesn't slow down as the data set grows.

III.III Project Scope

With an ambitious project like this it's been important to have a well-defined scope as the possibilities for execution are very broad. Originally in the scope was having the application served on the web, this would have allowed for cross platform and mobile use. However this was less relevant to the research point of the project so I refined to scope to focus on the core parts and functionality of the program. The scope includes the creation of a tool to collect data, and a product to use the data. The product should produce a text translation output, taking no regard for how close to real English the sentences was, just an exact translation of the users' actions. Whereas

the data collection tool should allow myself, and other users to record and collect additional signs towards the data set. Alternatively the user could record new data for already existing signs to increase the number examples for each sign.

IV Background

IV.I Sign Language

Sign language is a form of communication that encompasses hand gestures, body language, and facial expressions[5]. Sign language is often the predominant language used by deaf individuals of hearing impairment[6][7]. Sign language is not universal and differs between countries, regions and even towns. As an example of this there is a British Sign Language(BSL) and an American Sign Language(ASL). Other examples of localised sign language include New Zeland Sign Language(NZSL), French Sign Language (LSF) and Greenlandic Sign Language (DTS). While ASL and BSL do share similarities, they ultimately hold more differences. A comparison between the two languages found that only 31% of signs matched each other[8]. Sign language is therefore very broad and many of its languages stem from completely different origins.

The World Federation of the Deaf(wfdeaf) reports that worldwide there are 70 million deaf individuals that use 200+ different forms of sign language[9]. At the time of writing, census.gov[10] estimates there are 7.79 billion people on the earth. If 70 million of those are deaf then that equates to roughly 0.90% of the population being deaf. the Office for National Statistics(ONS) estimates that there is a rough population of 67m people in the UK as of 2020[11]. Using the data from the British Deaf Association which suggested there was 150,000 BSL users in the UK [1] we can safely say that roughly 0.22% of the UK population use sign language.The Royal National Institute for Deaf People (RNID) reports from the ONS that 12 million adults have a hearing loss of greater than 25 dBHL[12]. This reinforces the need for accessibility for the population of the world and people in the UK.

As the use of technology increases to grow across the globe it's important that our world is inclusive of that 0.90% of the population that are deaf, and the many more with hearing impairments. Accessibility products are limited

and often expensive. Creating a more inclusive world of people who's primary language is a form of sign can help better connect our world. Video calling software saw an increase in downloads of up to 30x higher than the previous years quarter average, during the week commencing March 15 2020[13]. This is around the time that lockdown was introduced to the UK. an increased demand for private and business video calls as loved ones and workers could not meet each other. The accessibility of most of these video calling apps is limited live captions or transcriptions of conversations but offer no way for sign language communication[14].

IV.II Capturing Data

One method of extracting data from our camera would be through the use of bounding boxes. Bounding boxes could either be manually drawn or acquired through deep learning as demonstrated in the paper Self-Taught Object Localization with Deep Networks[15]. Self-taught localisation was able to produce results comparable to manual tasking. Object detection allows a trained model to extract data from a larger image through the use of bounding boxes[16]. Using this method for our model we would have to train the model on a subset of each image, defined by the bounding box. A proposed method of achieving this for static hand gestures suggests the following technique: Input, Track hands, Feature extraction, Machine learning and, Classification[17].

Googles MediaPipe is an open source solution for implementing machine learning. It offers solutions for object detection, face detection, hand tracking and much more[2]. One solution it offers is the holistic package. The holistic package allows developers to efficiently implement the sampling of landmark positions for hands, the face, and body pose within an image [Figure 1]. The implementation of the holistic package allows extraction of key points of an individual. There are 543 landmarks available which are distributed as shown in

Figure 1: Examples of MediaPipe Holistic being used.

Identification	Left hand	Right hand	Pose	Face
Landmarks	21	21	33	468

Table 1: Holistic landmark totals [18]

For both hands, and the face, each landmark is comprised of three datapoints. These are an X, Y and Z. The pose also features an additional point of data, a Bool for visibility[18]. The MediaPipe Holistic package offers a lightweight solution and reports frames rates of up to 20 on the Galaxy S9+ and 17 frames on the 15-inch MacBook Pro 2017[19].

Overall through the research I have been able to carry out for data extraction I have discovered two options. The first of the options is to generate bounding boxes, and then train my model on the by feeding in subsets of images defined by those bounding boxes. The other option is to use the MediaPipe package to extract key point values from the images. Storing keypoints would require less data and be easier to process but would mean

we are unable to view the original images at any point so it's hard to validate the correctness of data after it's stored. Storing images would be much slower, and mean that users may struggle to hold larger data sets and record their own data. The major benefit of storing the keypoints is that the user only has the data they need. those keypoints define the human we're seeing down to the finger. This is therefore the best option to use.

IV.III Machine learning

Action recognition and sequential classification is not a new topic.Therefore this project is helped by the research material available to help make decisions on this project. Recurrent Neural Networks (RNN) and more specifically, Long Short Term Memory (LSTM) [Figure 2] are types of neural networks that can be used to solve machine learning problems for time-series and temporal data[20]. Previous work has shown that LSTM models work significantly well with video classification[21]. While RNN's do retain data through time they struggle to retain information throughout the whole sequence. LSTM however excels at learning the long range dependency between the initial input and the output label[20].

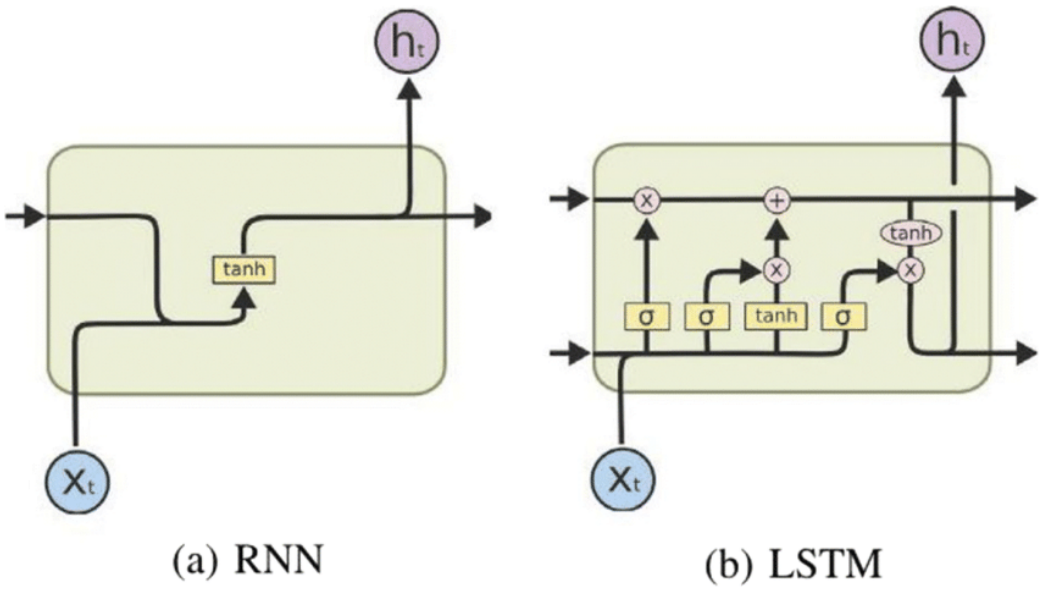

Figure 2: The general schema of a RNN unit versus a LSTM one (adapted from Olah 2015)[22]

Tensorflow[23] conveniently has a range of tools that can help us to complete this project. The sequential model offers a solution to build a custom model out of layers[24]. Building the model *sequentially*, will allow us to test and tweak the model to better work the type of data we have. Tensorflow offers a selection of layers that use the RNN architecture[25] including the LSTM layer[26]. PyTorch[27] is another alternative that could be used to implement our machine learning in python. PyTorch like Tensorflow offers the ability to build a sequential model[28] that has layers such as the LSTM[29] available to it. Pytorch performs slightly faster but Tensorflow performs slightly more accurately. Overall the impact of package selection is negligible[30].

IV.IV Comparison to other products

Ideally I would compare my project to other available products on the market. However it has been virtually impossible to find any commercial or public

software available. There are a plethora of applications that visualise sign language from speech or text, but a complete lack of sign language to text[31]. Articles from the past few years do however talk about supposed applications being on the brink of release but now, further down the line, there is no trace of them[32]. Despite these facts there are a handful of personal projects and research papers that exist and we can attempt to make a comparison to them.

The project *ActionDetectionforSignLanguage*[33] offers an example application that implements sign language translations. This implementation offers a test and train approach. The application allows the user to specify a set of labels with the amount of sequences to record for each. The application records all of these before moving directly onto real time classification. The application offers the user no way to record their data-long term, or add to existing data sets. However the application does provide a good demonstration as to how powerful MediaPipe can be. The application, like the research carried out previously, suggests that Google MediaPipe and Tensorflow would be candid packages to use.

The paper, *sign language recognition with long short-term memory*[34] provides an explanation and example implementation of an LSTM based model for sign language recognition. Carried out on two data sets. The LSTM model managed accuracy's of 0.856 and 0.62 respectively. Their implementation was tested on a large data set and showcases the power LSTM provides for temporal data. Another paper further emphasises the power and also implements 100 signs, but on Thai Sign language and received accuracy of 0.97 on test data[35]. The work these projects describe and detail will allow our project to benefit from tested decision making.

V Approach

The projects approach was divided into three main sections. The sections were research, foundations, and the development. The initial research was carried out so that I would have a stronger set of knowledge about the topic I was undertaking. During this time, I explored many different options about how to move forward with the project, an example avenue being the automatic calculation of bounding boxes[16]. Once I had formed a clearer path of how I would undertake my project, I moved forward and started preparing my work environment, created the foundations of my program. I created the files and built their respective user interfaces, which gave me a great starting position to better carry out work on the programs functionality: machine learning and predictions.

V.I Research

Research is the first part of my approach to this project. Getting a good understanding of the methods and technologies available to solve my issue would save me time in the long run and give me a more concise program at the end. I therefore split my research into three main areas. Image capture, object detection and sequence prediction. These were the three main things that I believed to be key to creating a good product.

V.I.I Image capture

To begin with I researched image capture. Having not used python before to process images or videos from an input device I knew this would be a valuable item to research. Researching this topic brought me to the package OpenCV[36] which seemed to be the most used and supported form of capturing webcam data into python. OpenCV contains lots of documentation[37] and support for python so I believed it would be the right selection. Later, this was also reinforced as Googles MediaPipe had direct support and

snippets for this library.

V.I.II Image detection

Object detection was the next thing that I would need to conquer. How was I going to extract the object, person, or hand, from the image I was capturing? To begin with research pushed me to use a manual masking tool to record the bounding box for the important information of the frame.[Figure 3] From my research in other papers, I believed that a form of extracting the key area of the image, hands and face would be the route I needed to take[38][39][40]. Luckily a couple of days later into research I came across Google MediaPipe which can recognise human features from an image[41]. This would prove as a crucial breakthrough and be the new route that I used for extracting my features from each frame. The Google MediaPipe Holistic package[18] allows the extraction of face, pose and hand data directly from an image containing a person. This was exactly type of data I would need. It was also much more lightweight to store this type of data than whole images and their bounding boxes.

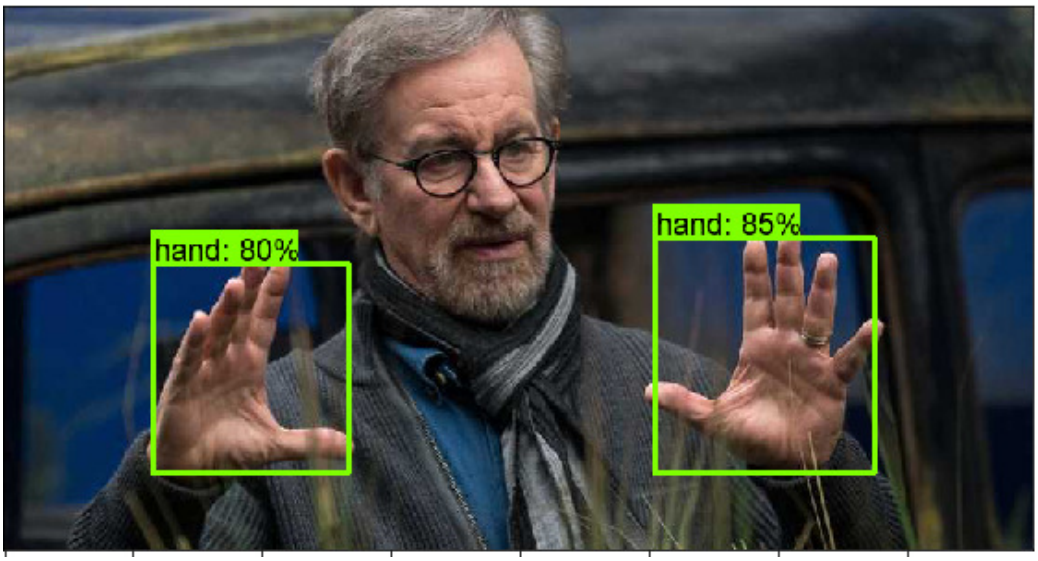

Figure 3: Example of bounding boxes displayed over hands[42].

V.I.III sequence prediction

The final part to research would be the types of models and layers available to complete the machine learning. Previously when completing static image classification tasks, I used CNN's which feed forward, this would not make a real utilisation of the sequence data. While researching I discovered Recurrent Neural Networks(RNN) which would be much more suited to the task.[43] The RNN is ideal for processing temporal or sequential data as opposed to the CNN. Furthering this I was able to research about the different RNN style layers available which would allow me to discover the LSTM layer onto which I would base my model from[25].

V.II Foundations

Creating the foundations of my project in advance of creating any of the core functionality would allow me a more robust and accountable working environment. To begin with I set up tools and procedures I would need to document and control my project. I then moved on to creating frameworks, and basic parts of my planned final product. This meant creating interfaces, installing dependencies, and dividing the project up into functional parts waiting to be created.

V.II.I Setting up the environment

Setting up the environment for my project was quick and simple. I was going to be working using Kanban[44] style methodology using the boards on Trello[45]. On Trello I would be able to lay out the tasks ahead of me and drag them in whenever I was ready. This contrasts to the usual Agile methodology where work is done to a stricter timeline, in sprints. As a single person working on this project, I felt this was a more appropriate development cycle that allowed me to make drastic changes without massive development cycle costs. The other main environment that needed to be

setup was Git[46]. Git would allow me to access older versions when needed and create useful branches for mid operation R&D testing. I used GitHub[47] to setup a private repository and used Git tortoise[48] to commit and push change.

V.II.II Creating the User Interface

After having decide the rough flow of my program and setting up my development environments I decided the next step would be to build the foundations of the programs I was building. This meant creating the user interfaces, they could then later be hooked directly into the more functional part of the program as and when that was completed. The user interfaces would be made in Python so that no middle point would be needed to send data between the different parts of the program. The user interfaces would be developed first, using a library I familiarised myself with for the project[49]. The interfaces followed the rough design patterns I had previously drawn up of how the application should look.

V.III Functionality

The main uncertainties for the program's functionality were the building of the model, and prediction of the sign. I knew the model would need various levels of tweaking before it worked ideally on the data set I was giving it[50]. In the functionality stage I allocated plenty of of time for the fine tuning of model parameters and for fine tuning the method of sampling the sign prediction.

V.III.I Building the model

Building the model would be a complex task of trial and error. Machine learning models can be made up of many different layers, with each one contributing in a different way. My goal here would be to try different combina-

tions based on the research I had performed and the documentation available for each layer to see which combinations would have the best outcome for the data I had provided. I would need to account for over fitting, the randomness of each model, and consider items like the batch size and epochs to use. As previously mentioned, I discovered a handful of RNN layers that may be useful for the project[51].

V.III.II Predicting the sign

Predicting the sign should seem easy as It's just feeding in a sequence to the already built model, but I would need to find the right solution to do so. By this I mean how often to make a prediction. Do we do it every frame, with the previous 15-frames, or every 5-frames?. My initial approach to this task was to take a prediction every frame and then average from the past predictions to get a frame. This method was unreliable and ultimately move my focus to creating a prediction every set time. Testing these different ideas would help me get a better understanding as to how the model makes its predictions based on the sequences it had available.

VI Specification And Design

VI.I Data Collection

VI.I.I User Perspective

Data collection is one of the main features of the solution I have created. It allows the user to record new signs and add additional signs to the data set which can be used by the translation tool. This process therefore needs to be intuitive for the user as the data they may record here will have a direct impact on the translations they receive later.

The user can collect data conveniently through the tool I have created. This tool features a user interface that will allow the user to specify the type and quantity of data they are collecting. When the tool is loaded the user will see the following items. At the top of the window a read only text box that denotes the location data will be saved. Below this is an area that will show the user the feed from the web camera. The user will see an outline of their face and hands when they are visible in the frame. Under this main area is a set of two text boxes that can be used for the user to specify the sign they wish to record, and to the right, the quantity of sequences of that sign. Continuing down are there is a button which is used to begin the sequence recording process. Finally, there is a label that is used to display the programs status to the user. The user will see this change once they begin recording[Figure 4][Figure 5].

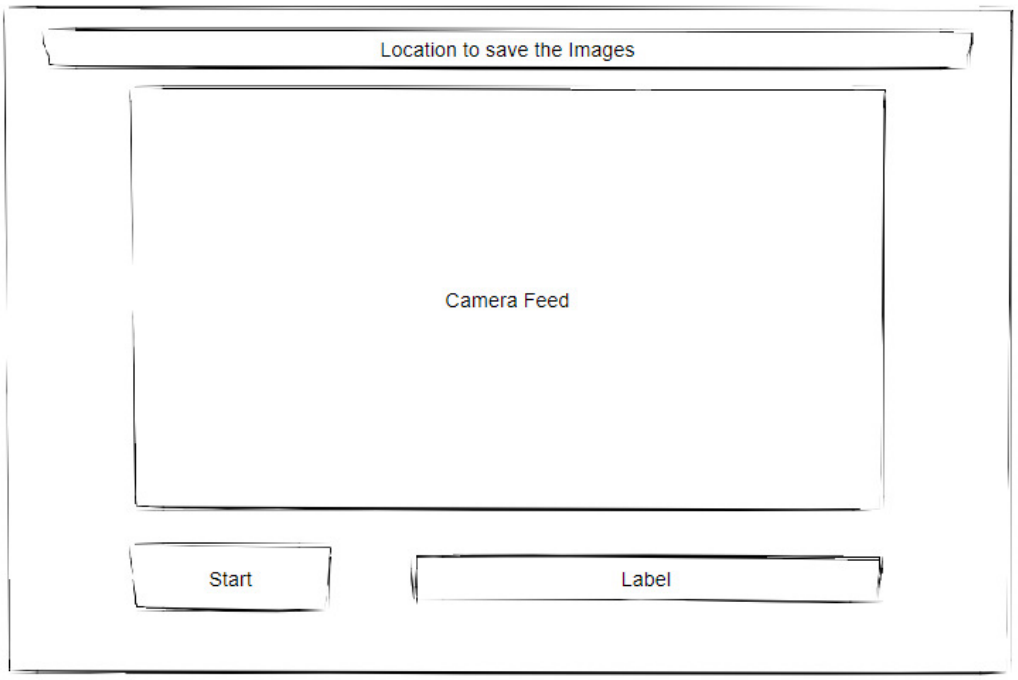

Figure 4: Initial Design of data collection tool.

VI.I.II System behaviour

When the system is first loaded it will try and create a hook to the code specified webcam. If no camera is found the system will not open. Once a hook is made the image box will continually be updated with the image of the webcam. If the image contains a recognisable face, and or hand, they will be drawn as an overlay of the image shown to the user[Figure 5]. The system will continually perform this task until the user fires the start recording event. On each frame loaded the program will check to see whether an event has been called which move the program into its next state.

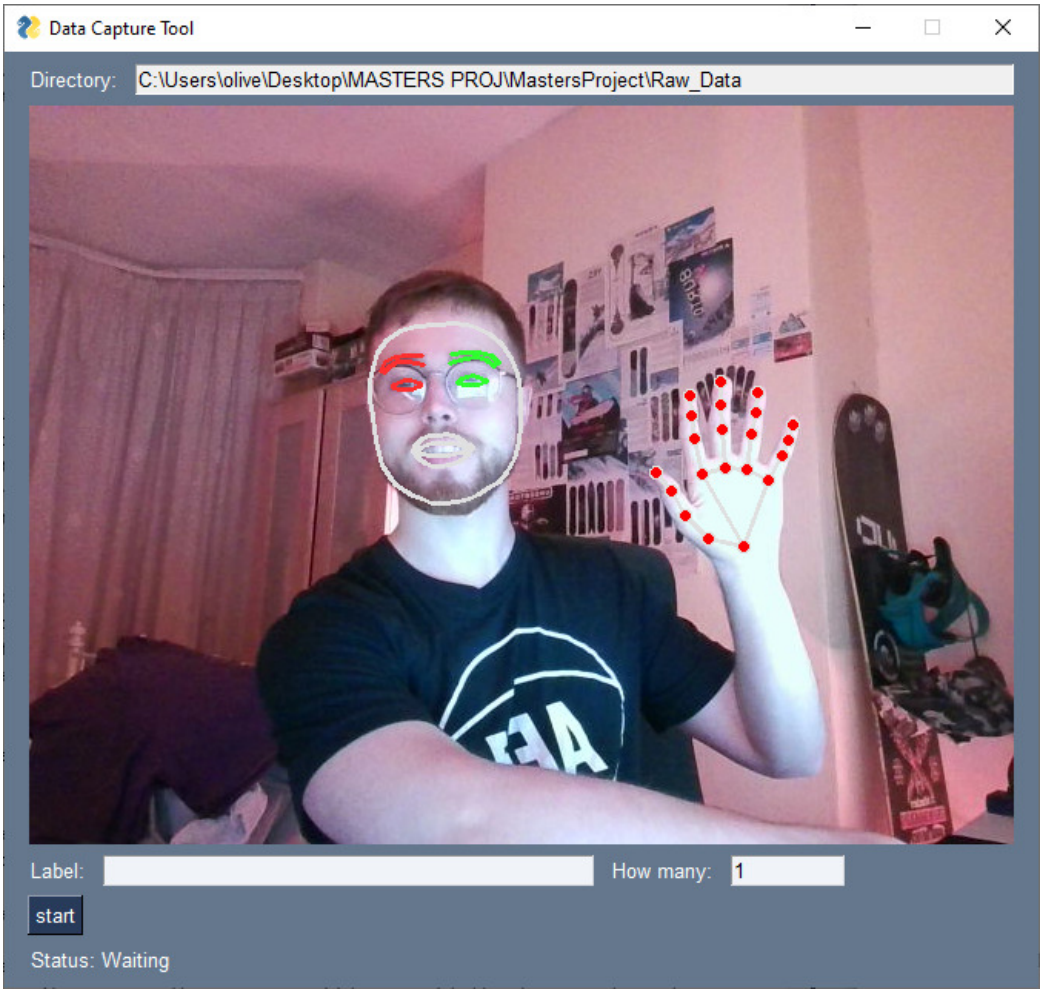

Figure 5: Implemented User interface of data collection tool with MediaPipe overlay.

When the event is triggered the label and quantity that the user specified on the interface will be loaded and used as variables to specify the type and number of sequences to be taken. The loop of sequence taking lasts for 15 frames, which are displayed to the user as a counter. Once the 15 frames have passed the user will see this change to a waiting phase, where 10 frames of grace will pass before recording starts again. The system will store the information gather(explained in Data Storage). Once the specified quantity

of sequences have been taken the program will wait for another start event. If the user exits at any point the program will close.

VI.I.III Architecture

The architecture of this process is very simple. All of the code that is accessed in this system is stored inside of the Capture_Signs.ipynb file. The file has a few external dependencies which form its main architecture. These dependencies are: cv2 for image capturing, Mediapipe for overlay drawing, PySimpleGUI for rendering the interface and numpy for data storage and manipulation. Each package plays an important role in this operation and combine well together to create this product.

VI.I.IV Constraints

There are several constraints of the system currently. In its current state many items that could be considered settings are only available through modifications to the source code. Although they are relatively simple to change it's not very user friendly as they will need to know what they are doing. Some of these setting include the quantity of frames to be collected per sequence, it's currently locked at 15, the quantity of frames to wait between sequences, some users may want to increase this from 10 to allow more time to re-position their poses. Similarly, the webcam selection is done in the code. It could be more beneficial for the user to have the ability to select from a drop-down of all available devices. Also, if the webcam is being used, or has recently been used by another process the application may fail to open, being unable to use the webcam.

A final limitation of the program in its current state is that it's still in the format of a Jupyter-notebook[52]. This is fine for advanced users but would not be very useful for the public. At a minimum the program could be opened better by converting its contents into a standard python file or in the end game entirely removing the constraint by making it an exe.

VI.II Data Storage

VI.II.I User Perspective

Data storage happens twice across all aspects of the program. Firstly, data is stored when the user records sign sequences in the data collection tool. Secondly some data is stored for use in the final translation tool when the user chooses to create a new model from the data they have.

The user's perspective of the data being stored should be detached. Although data is being stored the systems automatically retrieve the data from the file structure that's automatically created. For that reason, the user should not need to have much of an overview of what's happening. However, the user may later choose to swap out or move files, so it is still important that its coherent. The user is presented with an easy-to-understand file structure of label name, sequence number, sequence files [Figure 6]. This allows the user to easily locate a specific sequence, or sign and remove it if needed. The user also does not need to work about naming conventions and the numbered folders and file names are automatically done to make life easier for the user

The two other files are the label map and the model. These files are stored in the root directory and code directory respectively. The user does not need to interact with these files unless they are swapping them for others, which means they likely have a more advanced knowledge of the system anyway.

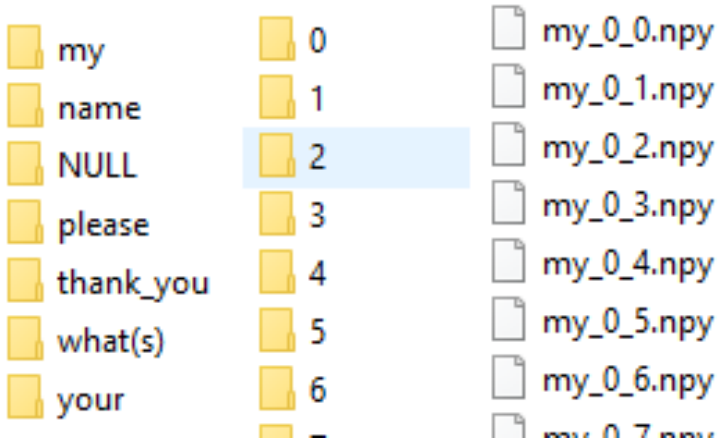

Figure 6: File structure generated and used by the data collection tool.

VI.II.II System behaviour

The system behaves very autonomously for storing data. For the data collection tool, the data begins being stored when the user fires the start recording event. The system will then look at the label the user has requested to record and see if it exists already in the data area. If the file does not exist it will make the folder and then for each sequence create the sequence folder and store each frame, as they care captured. Cleverly if the system detects that the file already exists for the label it will look inside that folder to determine how many sequences already exist. It will then begin counting from the maximum folder in there. This was the data can be collected on top of the existing data.

The data that is stored consists of arrays of landmark points that were captured from the users faces and hands. The system stores these values as numpy arrays. These arrays are much smaller and easier to work with than storing actual images.

The other files the system generates are done during the model training. When the data is read in for model training a set of labels is generated and saved as a text file. This is so that the translation app can later translate the prediction results from one hot encoding to a label value. Once the model is finished being built it is saved inside of the code folder, again to be used by the translation system.

VI.II.III Architecture

The architecture of data storage is set across two files. Firstly Capture_Signs.ipynb houses all of the structure to save the image data into the data area. It uses two python packages, OS and numpy. OS is used to do directory browsing, to create new folders and check for existing ones, and numpy is used to create and save the data to the disk. The label map and model storage code are

both contained in the Model.ipynb file. Here pythons default functionality can save the text file and the Tensorflow package can save the model. The architecture for saving the data is pretty simple but is cleverly designed to keep autonomy and hassle away from the user.

VI.II.IV Constraints

One constraint of the data storage is the file size. Although each frame in a sequence takes up $\tilde{1}$3KB, when each sequence holds 15 frames $\tilde{1}$95KB and you have 100 sequences for each label, things start to add up $\tilde{2}$0MB per label. If you were to have say 100 possible signs you are then looking at data of $\tilde{2}$GB. Of course, in the future it would be ideal to have the full dictionary here. Data would better be compressed at the time of storage and uncompressed when its used to train the model. One other slight constraint is that the label map, model, and data are all stored separately. Since these files are all generated by the program it could be beneficial to store them together. This means it would be easier for a user to swap out their entire data set with a new one more easily as they would not need to find and replace each file individually.

VI.III Building the model

VI.III.I User Perspective

The building of the model stage is slightly less user interactive than the data collection and the translation tool. Currently this is done as a Jupyter-notebook script and prints its results as outputs in the output/console area.

The user can run the script by pressing run all in Jupyter-Notebook. The user will then have to wait for a prolonged period before they get a result. The user will first wait through all the epochs of learning, they can also leave the program to run during this time. Once the user has waited for this, they will be able to see the following information. The accuracy of

the trained model, the parameters, a confusion matrix [Figure 7]. The user should not be interested in the parameters but can check how well the model was trained via the accuracy result and the heat map. The user WILL be given guidance elsewhere on what constitutes a good model. The user can see which predictions were wrong and what it predicted them to be, such as predicting 10 pleases' as thank you.

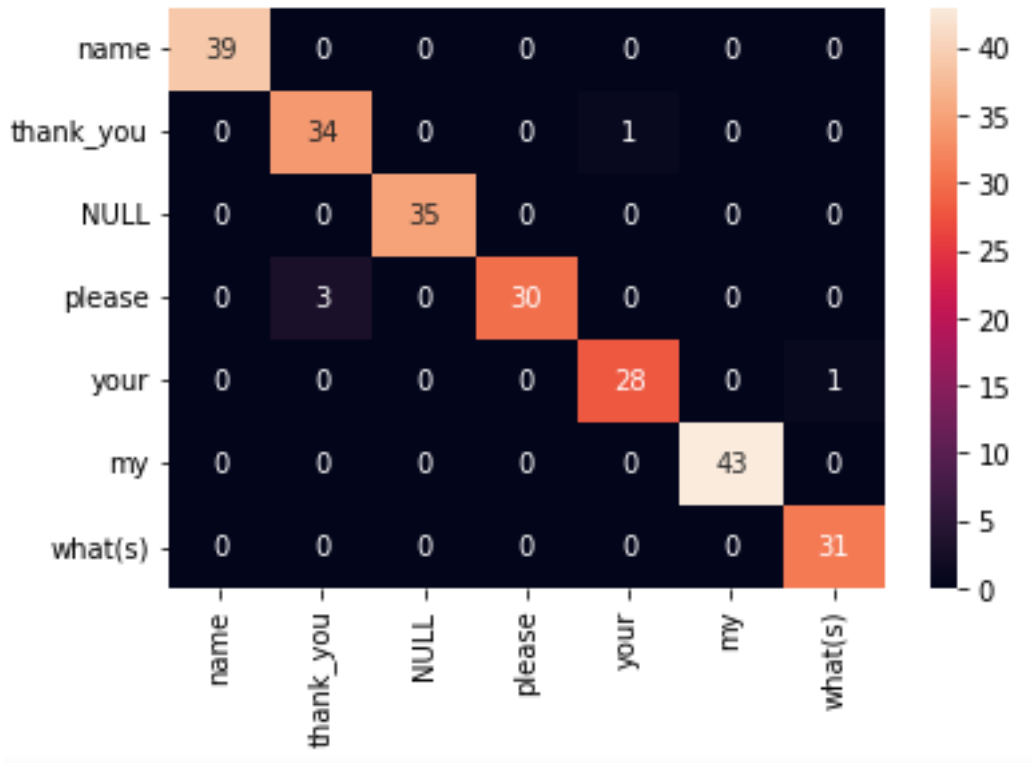

Figure 7: Confusion Matrix generated by the model.ipynb.

VI.III.II System behaviour

When the system is launched it will begin by importing all the data from the program. It will search through all of the folders currently located in its specified Raw_Data path. The data will then be brought in and stored as arrays of arrays of arrays, label, sequence, and seqdata. The data will then

be divided into sets for the model to train and test on. Once this data has been divided the model will begin to build itself. The time taken depends on the amount of data available; other options like epochs and batch size will affect these too. Currently these are hard coded but could be more dynamic to account for the change in data sizes.

The model will train for however many epochs are specified in batches of the specified batch size. Once the training is complete information about the model is output for the user. This informs the user how good their model is, accuracy and confusion matrix. The model is then saved by the program so that it can be used in the final product, the translation application.

VI.III.III Architecture

The architecture of this file is relatively straight forward like the other files. This code is located inside Model.ipynb. It uses data from the ..Raw_Data path. Outputs the file model.h5 locally, and outputs the file ..labels.txt. The Model.ipybn file uses the Sklearn and Tensorflow python packages to perform it's machine learning. Tensorflow.Keras is utilised multiple times for items such as the model, its layers and organising the training data. The overall architecture flow of this system is to import the data, process it and output a model. This system is the middle component of the larger system.

VI.III.IV Constraints

The main constraint of this system is that it is not very user friendly. Currently the user needs knowledge and guidance to understand what constitutes a good model. This could be changed to have threshold values that will re-run the model process until a level of accuracy is met. Another constraint is the user could believe that an accuracy of 90% to be acceptable with lesser knowledge of how the confusion matrix works. The accuracy could be 100% apart from one label which could be 0% and have all its results predicted wrong. Another constraint of this is an information overload. The stand

user does not need to, or likely know what an epoch is. This information should be hidden from the window. A user interface, or console could be better suited to give status messages and predicted times of completion. One final constraint here is that the user could continually get bad models for a long time. When the program has more labels and is more complex it could take a lot longer to train the model. When the user adds more data, they will need to retrain the model. Due to the randomness, and dropouts the model could fail to be good for a few iterations of building. This could lead to very long waits for the program to be more usable. Implementing a system where the current model can just be progressed with newer data, retaining its old training would be better for the user.

VI.IV Translating signs

VI.IV.I User Perspective

The data capture tool is the final piece of the puzzle. This is the actual product that gives the user a translation from the signs they are doing.

When the program is loaded the user is greeted with a simple screen. The screen contains three items. The first item is the camera feed. The camera feed is the same as the one from the data collection tool apart from there is no overlay this time. The user does not need to see the frame of the pose they are doing. To the right of the page is a larger text area. This text area is where the outputted translation is placed. The text area is editable by the user so they can remove items, add extra text if need and copy and paste out their translations. Finally at the bottom of the page is a counter. The counter informs the user, like the data collection tool, whether they're in the capturing or grace period[Figure 8].

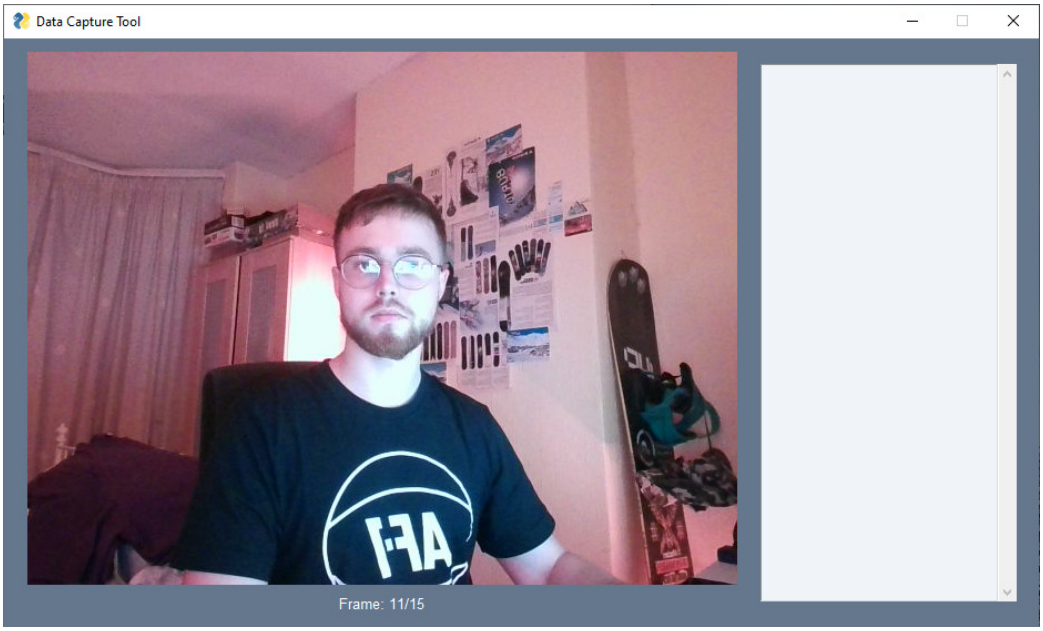

Figure 8: Implementation of user interface for translation application.

VI.IV.II System behaviour

The system works in a continuous loop. Before the loop starts the model and label map are loaded into the program. Once they have been loaded the system hooks into the camera like before and begins its main loop. The system collects a frame and stores it in a queue until the queue is full. This is when 15-frames have happened. As the frame data is collected its converted into the same format as the data that was stored and used to build the model. With 15 frames a prediction is then made by passing the queue to the model. The model provides an output, which is converted to a label and presented to the user in the translation window. The system will then wait before collecting another set of frames.

The system is set to recognise and not display a null pose, or a translation that is the same as the previous one. A null pose is modelled after a sequence of images in which the hands are not used. This seemed an ideal way to present a “no prediction” option. The system will repeat this entire process

until the application is closed.

VI.IV.III Architecture

The architecture for this part of the program is inside of the file Translator.ipybn. The program makes use of the external python packages: cv2 for image processing and capture, PySimpleGUI for the user interface, MediaPipe for collecting the landmarks from the images, numpy for data manipulation and modification and finally Tensorflow for importing and predicting with the model.

This tool is the main part of the project and utilises all the data that has been collected and generated from the previous parts. The file takes as inputs all this data and outputs text directly onto the user interface to be used. Once again the architecture could be slightly improved here by converting the project before any release into at least a python file but ideally an .exe.

VI.IV.IV Constraints

There are a few constraints of this system. The first constraint is that the user cannot resize the window. Unfortunately, this has not been added in, so the user is stuck with the default smaller window size. This also means that the quality and resolutions of better cameras is not fully utilised. Keeping the resolution low, however, keeps performance higher currently. Another of the constraints this program has is that the application begins to attempt translations as soon as it's opening. There is no option to start or stop the translation. This would be something to add to the program for better usability.

Another constraint but also an item that's addressed in future work is the way in which signs are translated. As previously said the signs are translated one at a time with a small waiting period between them. It would be better for the program to do it more fluidly as the user strings signs together etc.

However, this is a current constraint by my knowledge and the time I had left to explore more about this option.

VII Implementation

VII.I Collecting data

VII.I.I Interface

The interface is implemented using a python packaged, PySimpleGUI. When the program is initially run, the packages window function is called with an argument of the layout to be used. The layout is defined in a function called get_layout(). This function returns a list of lists, where each sub list is acting as a row in the user interface. The layout is made up of the following PySimpleGUI elements: Text, Image, Button, and Input. Elements have the attribute k or key set to allow the program to access them later[Figure 9].

```
def get_layout():
    layout=[
        [sg.Text("Directory:"), sg.Input("C:\\Users\\olive\\Desktop\\MASTERS PROJ\MastersProject\\Raw_Data", size=(81,None),
        [sg.Image(filename='', key='image')],
        [sg.Text("Label:"), sg.Input(key="lbl"), sg.Text("How many:"), sg.Input("1", size=(10, None), key="lbl_record")],
        [sg.Button('start')],
        [sg.Text("Status: Waiting", k="status", size=(50, None))]
    ]
    return layout
```

Figure 9: Code implementation of the data collection layout.

This applications main logic runs in a loop for each new frame captured by the camera. On each of these iterations a call is made to the window, indexing the key for the interface element and the update function is called with the argument of the new image variable[Figure 10]. Similarly, the update function is also called with the keys of the text elements to update the user as to the status of the program. The variable count is increased for each frame until the threshold of FRAMES_PER_VIDEO is met. This controls whether the frames are to be saved. Once the threshold is met, a grace period is achieved by incrementing the count variable until the threshold of FRAMES_PER_GRACE is met.

```
imgbytes = cv.imencode('.png', image)[1].tobytes()
window['image'].update(data=imgbytes)
```

Figure 10: Code implementation of updating the Image element with the next frame.

The interface also listens for the user clicking the Button element. On each iteration event, values = window.read(timeout=20) is ran which collects any events which took place. This allows the program to switch the variable, recording(Bool) to True and begin the capture process.

VII.I.II Capturing key points

The program collects data about the position of the users face and hands. This task is achieved using Google's MediaPipe. The implementation uses the mp.solutions.holistic package. This allows us the option of collecting face, hands and pose data. Firstly, we define two variables mp_drawing, mp_drawing_styles, to shorten access to the package properties. We then use a with statement to cleanly use the package and handle any errors. On each frame we call mp_drawing.draw_landmarks(), with the arguments of the image and the type of holistic we would like to draw[Figure 11]. In my solution I opted to only implement face and hand data. This displays the key points to the user.

```
mp_drawing.draw_landmarks(
    image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
mp_drawing.draw_landmarks(
    image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
mp_drawing.draw_landmarks(
    image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS,
    landmark_drawing_spec=None,
    connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_contours_style())
```

Figure 11: Code implementation for drawing landmarks over camera feed.

Next we have to extract those key points. On each frame of the program that is recording a call is made to the save() function in the numpy Python package. The arguments are the file name and the data to save. The data is generated by calling extractResults(results). This takes all the MediaPipe

data as an argument and converts it into our desired format. First a new empty list called frame_data is created, a list of the default sizes for each holistic is also defined. Over the desired holistic the data is extracted and added to the frame_data. A check is made; if res_type is None: this will add a list of 0's of the right size to the frame_data[Figure 12]. This ensures all data is the same size. The data for each point is made up of an X, Y and Z value. This is then returned and saved as mentioned before.

```
def extractResults(results):

    #how many landmark points per landmark type
    sizes = [63, 63, 1404]
    frame_data = []
    for i, res_type in enumerate([results.left_hand_landmarks, results.right_hand_landmarks, results.face_landmarks])

        #Creates 0's if there are no landmarks avalable (all data needs samesize)
        if res_type is None:
            frame_data += np.zeros(sizes[i]).tolist()

        #concat the landmark data
        else:
            for res in res_type.landmark:
                frame_data += [res.x, res.y, res.z]
    return np.array(frame_data)
```

Figure 12: Code implementation of converting Holistic data to desired format.

VII.I.III Output

The program manages to autonomously name and save to the correct location the outputs of this program. The program makes use of the OS package that is internal to python. When the variable recording is set to true, for the currently user defined label a path is generated in the location variable to where the file should be saved. os.makedirs(location, exists_ok=True) is called to attempt to create the directory for the label if it does not exist. The variable initial_video_count is initialised by calling the len() function on os.listdir(location) which lists all directories. This allows us to know how many sequences already exist for this label. For each sequence being filmed we use os.makedirs() again to create a folder inside the label folder with the sequence number. Using initial_video_count as the base, and a counter video_count that is incremented each new sequence, we can accurately num-

ber the folder. The files to be saved are named using a concatenation of this information like so: Label_SequenceNo_FrameNo.npy. here the FrameNo is the counter that is also used to switch between recording and grace periods.

VII.II Building the model

VII.II.I Importing the data

Importing the data is the first step when building the model. The function import_files() returns two lists, X_data and Y_data. The identical indexes of these lists correspond to each other, so X[1] is the X data for Y[1]. The data is imported by looping through each Label, Sequence and then Frame. Inside the loop for each video, an empty list frames is initialised to hold all the frames for that video. The Y_data value is also initialised with the name of the folder (the label). The function listdir returns items in an arbitrary order[53]. The frame number is extracted from the title by using replace() to remove the .npy, and split() to access the frame number. This is appended to the start of each frame array. A sort is performed sorted(frames, key=lambda x: (int(x[0]))) [Figure 13] to order them based on that newly appended frame value.

The frames variable is now appended to the X_data, but first removing the frame number from the start.[Figure 13] This process is repeated by the loop, once again appending a new Y_data value and X_data frame set, until the loops are finished and the data returned by the function. The data is then split into a test and train split using the sklearn.model_selection.train_test_split() function.

```
    frames = sorted(frames, key=lambda x: (int(x[0])))
X_data.append([x[1:] for x in frames])
```

Figure 13: Code implementation to sort based on the first item in the list, and then append all data but that value.

VII.II.II The Model

The Model was the hardest part to implement for this project. The model took a lot of R&D before I was happy with the final product. Previously the model would over fit but with the implementation of items like dropout it performed much better.

The model is implemented inside of the function evaluate_model(). The variable model is first initialised as a sequential model using the function Sequential() from tensorflow.keras.models. The model is next subject to the addition of several layers. 6 of which are hidden, after the initial input. The layers used are LSTM, Dense and Dropout[Figure 14]. These layers are all part of tensorflow.keras.layers. Three LSTM layers are added using model.add() with LSTM() as the argument. The first LSTM layer defines the input shape using input_shape=(15,1530) where 15 is the sequence length and 1530 is the amount of data points for each frame. The LSTM layers are initialised with the respective sizes 64, 128, 64. After experimenting with different sizes these worked well over many training tests. The final two LSTM layers also feature dropout using the parameter recurrent_dropout=0.1 and 0.05 respectively. This dropout helps the model top over fitting.

Three more hidden layers are added to the model. Dense(64), dropout(0.1),Dense(32). The final hidden layer is added to the model with this line model.add(Dense(Y_size, activation='softmax')). This sets the output to be the size our Y space, using softmax to give a probability for each Y value. Now that the model has been fully implemented it is compiled using model.compile() and finally fitted using model.fit() where the arguments are the X and T training data, 300 for the size of the epochs and 16 as the batch size.

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation="relu", input_shape=(15, 1530)))
model.add(LSTM(128, return_sequences=True, activation="relu", recurrent_dropout=0.1))
model.add(LSTM(64, return_sequences=False, activation="relu", recurrent_dropout=0.05))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(32, activation='relu'))
model.add(Dense(Y_size, activation='softmax'))

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
```

Figure 14: Implementation of the machine learning model.

VII.II.III Evaluative Techniques

Several evaluative techniques have been added to the program to explain the results of the created model. The evaluative techniques take place inside the function evaluate_model() which is called to create the model. The evaluation happens after and while the model builds. The first implementation of an evaluation technique is the accuracy. model.evaluate is called passing in the X and Y test data. The return of this function is stored inside the variable accuracy, and denotes the percentage of successful predictions.

Another technique used is to produce a confusion matrix. Firstly the X_test data is fed to the model for prediction and stored in a Y_pred variable. Then we use Sklearn.metrics.confusion_matrix to create a confusion matrix. using the package seaborn we can use it's heat map function to generate a visual output of the confusion matrix. the heat maps axis labels are mapped to label_map.keys() which allows us to see which labels were incorrectly predicted and as what. Finally using model.summary(), we are able to see the parameters of the model we had generated.

VII.III Translation

The final component of the project is the prediction application. The prediction application has similarities of implementation with the other two items; therefore, I will cover the unique parts. The implementation of this interface uses two Column elements to divide the page. One contains the Image

element and Text element while the other contains the Multiline element. The window is rendered the same as collecting data. When the program is launched a hook is made to the webcam using cap = cv.VideoCapture(1) to target a webcam device and while cap.isOpened(). A variable to hold the frames is created by current_frames = FrameQueue() where FrameQueue is a custom class. The model is loaded using tensorflow.keras.models.load_model and the label map is read in too. On each frame a clever check and action is performed if current_frames.enqueue(image_data).is_full():. The enqueue function adds the data and returns the new object which is immediately checked for being full by .isfull() When .isfull() returns true a call is made to the function predict(). Predict() receives as arguments the 15 frames, the model and a copy of the label_map. Model.predict() is called to receive a result in the form of one hot encoding which are then translated by indexing the label map before returning the predicted result. A check is performed against the result to see if it's NULL or equals the previous translation. If neither are the case the translation variable is appended too with the latest translation. Using the window[key].update() method we are able to push the translation string straight into the Multiline element. The Queue is then cleared and this process starts again. Like before a counter is used to switch between the 15 frame recording and 10 frame grace periods[Figure 15].

```
if current_frames.enqueue(image_data).is_full():
    #Predict the output.
    res = predict(current_frames, model, labels)
    #Print it to the screen.
    if res != "NULL" and translation.split(" ")[-1] != res:
        translation += " " + res
        window['translation'].update(translation)
    #Clear the fram queue and go to wait 10 frames.
    current_frames.clear()
    wait = True
```

Figure 15: Code Implementation showing the sequence of adding frames to the queue, making and displaying a prediction.

VIII Results and Evaluation

To evaluate the project, I will look at two things. Firstly, the performance of the model. The performance of the model can be evaluated by looking at its self proclaimed metrics on test data and the scores it achieves when being tested by a human. Secondly, we can evaluate the project against the aims that were created at the projects start. The speed and performance of the application is based on the system that its being used by. Running the project on a £300 laptop was probably about the minimum. Better results may be achieved by collecting data and translating on a computer that has a better processor and graphics card.

Overall after the findings and discussions that are carried out in the below two evaluations. I believe the project has done well. Meeting high statistical efficiency and also meeting, for the mass majority, the aims set.

VIII.I Evaluation by User testing

The products performance can be based on how well it is able to translate signs. The file that is capable of this is the model.h5 file. The model.h5 file was trained on a data set comprising of 7 labels, each with 100 sequences of 15 frames. The model was trained using 65% of the data, and then tested upon the remaining 35%. The model performed well with an accuracy of 97.95%. Overall, on this test data set the model failed to predict 5 sequences correctly. A heat map generated from the confusion matrix describes[Figure 7] the distribution of failed predictions. You would not expect to get 100% results as this may indicate that your model is over fitted to the data set you provided it with.

After receiving this excellent model, I performed some user testing. The user testing was performed by attempting each trained sign sequentially 10 times. The results were recorded and can be seen in Table 2. The outcome of the sign was recorded as either correct, wrong, or no sign. As you can see

most signs fair well. With Name achieving 100% accuracy, my receiving 90% accuracy and, whats and your achieving 80% accuracy. NULL also received 100% and is tested by no sign appearing when no hands are shown. The overall accuracy I achieved through my user testing comes out at 72.85%.

Label	**No Translation**	**Wrong Translation**	**Correct**
What(s)	1	1	8
My	1	0	9
Name			10
Please	3	4	3
Thank_you	1	6	3
your	3		8
NULL			10

Table 2: Results of user testing

The two signs that did not fair very well at 30% were please and thank_you. The system had a hard time figuring out which was which and often got confused with other signs. I could have generated a confusion matrix to demonstrate my own findings for incorrect translations but did not at this time. What is noteworthy about these results are that 4 of the 5 incorrect predictions from the test results involve thank you, and 3 of the incorrect predictions involve please. The two signs for please and thank you are the same for the hands. The mouth movement however mimics the word being signed. This may indicate why the product has a slightly harder time with these signs. Two possible solutions to this problem are: increasing the size of the training data or alternatively having them as the same sign with a contextual understanding of the sign by the user.

Overall, I believe that the model I have built to go with the product demonstrates its capabilities well. Performing well statistically on a small-medium sized data set with 98% accuracy and a real-world accuracy of 73% I

believe this demonstrated the capabilities and abilities of the program. There is more work to be done but this program showcases the projects power. The products performance.

VIII.II Evaluation by Test cases

Unit tests have been written and carried out by myself on the programs that I have created. The user tests are there to test the original three aims I had provided and the criteria I had listed to meet those aims. The three aims I have tested are: When the user performs a sign, a text translation is provided (tests 1-3), The application is easily adaptable to incorporate more signs and gestures (tests 4-6) and the application runs fast and smoothly (tests 7-9). All the tests that I carried out passed and the actual outcome was exactly as expected. The tests were completed at the end of the projects coding phase and therefore should be correct when coupled with the final submitted product.

I believe after carrying out the test cases I have written that the project does a good job to meet what it set out to do. The project passed on all written test cases and therefore satisfies the aims and their criteria that were set out in the project's introduction. The criteria for the second aim however do not satisfy its intent fully. The application is easily adaptable to incorporate more signs and gestures is the second aim. The reason I think this aim was not fully fulfilled is because the difficulty the user would currently have to train the model. The user is currently easily, thanks to the capture_signs tool able to capture more signs. However, it's very inconvenient and not easy to incorporate those into the data set.

Currently when the user would like to add their data, they would have to retrain the whole model. They would not and should not be expected to have the knowledge to understand whether the model they produce is good or not. This could jeopardise the system as users may be exposing them to sub par models and in which case, software. After evaluating the product

through the test cases and discovering this lack of aim meeting, I researched how to rectify this problem. Quite easily an option or script could be added that loads the current model and fit's it with the additional data. This would satisfy the condition of "easily adaptable" as it would be fast and convenient.

VIII.III Test cases

<table>
<tr><td rowspan="2">Test Number: 1</td><td colspan="2">Aim: When the user performs a sign, a text translation is provided.</td></tr>
<tr><td colspan="2">Acceptance Criteria(s): The application clearly displays the output translation to the user.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.
• Point the webcam at your self to show your torso and face.
• Open the application
• move hands below the screen before application opens.</td></tr>
<tr><td colspan="3">Steps:

• Wait until the status at the bottom says "Wait"
• Move hand up to but away from head in two finger salute position
• Wait until the status says "frame"
• Move hand towards head and back to start position within 15 frames
• Mouth the word name while doing so
• Move hands back below the camera</td></tr>
<tr><td>Expected: The box to the right of the camera feed should now contain the word "name" and nothing else</td><td>Actual: The word name is displayed in the multiline text area</td><td>PASS</td></tr>
</table>

Table 3: Test Case 1

<table>
<tr><td colspan="2" rowspan="2">Test Number: 2</td><td>Aim: When the user performs a sign, a text translation is provided.</td></tr>
<tr><td>Acceptance Criteria(s): The user can copy or export the translation from the application.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.
• Point the webcam at yourself to show your torso and face.
• Open the application.
• move hands below the screen before application opens.
• Complete Test Number 1 steps.</td></tr>
<tr><td colspan="3">Steps:

• Move the cursor over the text area
• Click in the text area
• Press CTRL+A
• Press CTRL+C
• Open Notepad
• Press CTRL+V</td></tr>
<tr><td>Expected: Text should be pasted into notepad with the same contents, “name”, as was in the application</td><td>Actual: The word “name”appears in the notepad.</td><td>PASS</td></tr>
</table>

Table 4: Test Case 2

<table>
<tr><td colspan="2" rowspan="2">Test Number: 3</td><td>Aim: When the user performs a sign, a text translation is provided.</td></tr>
<tr><td>Acceptance Criteria(s): The application make san attempt to recognise ”no sign” or false signs.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.
• Point the webcam at your self to show your torso and face.
• Open the application
• move hands below the screen before application opens.</td></tr>
<tr><td colspan="3">Steps:

• Leave your hands below the camera for 60 seconds.
• Wave hand into the frame briefly during the frame counter
• Bring hands back to rest</td></tr>
<tr><td>Expected: The application should render no words into thetranslation box. The application should detect that the movement is not a sign it knows.</td><td>Actual: Nothing is displayed in the box. If the hand is displayed for the full time sometimes a sign may appear. The program “attempts” would be a good definition</td><td>PASS</td></tr>
</table>

Table 5: Test Case 3

<table>
<tr><td rowspan="2">Test Number: 4</td><td colspan="2">Aim: The application is easily adaptable to incorporate more signs and gestures</td></tr>
<tr><td colspan="2">Acceptance Criteria(s): The application should be extensible and be able to detect action sequences other than BSL if needed.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Remove the file Code/model.h5
• Remove the folder Raw_Data
• Remove the file labels.txt</td></tr>
<tr><td colspan="3">Steps:

• Set up an external webcam.
• Run Capture Signs.ipynb
• Record 100 of the hand movement A of choice that is not BSL
• Record 100 of the hand movement B of choice that is not BSL
• Run Model.ipynb
• Wait
• Run Translator.ipynb
• Test hand movement A
• Test hand movement B
• Check prediction results.</td></tr>
<tr><td>Expected: The program should identify the hand movements correctly.</td><td>Actual:
The program correctly identifies the new hand signs that are not BSL</td><td>PASS</td></tr>
</table>

Table 6: Test Case 4

<table>
<tr><td colspan="2" rowspan="2">Test Number: 5</td><td>Aim: The application is easily adaptable to incorporate more signs and gestures</td></tr>
<tr><td>Acceptance Criteria(s): The user should be able to use different data sets.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• The user has two different model.h5 files
• The user has two different labels.txt files
• The files correspond correctly.</td></tr>
<tr><td colspan="3">Steps:

• Set up an external webcam.
• Run the program Translator.ipynb
• Perform a sign and check the output.
• Close the program
• Switch out the model.h5 file
• Switch out the labels.txt file
• Run the program Translator.ipynb
• Perform a sign that was not in the other model and check the output.</td></tr>
<tr><td>Expected: The model should correctly identify the sign. The sign in the second model is output correctly.</td><td>Actual:
When the model and labels were switched the translator seamlessly used the new files.</td><td>PASS</td></tr>
</table>

Table 7: Test Case 5

<table>
<tr><td colspan="2" rowspan="2">Test Number: 6</td><td>Aim: The application iseasily adaptable to incorporate more signs and gestures</td></tr>
<tr><td>Acceptance Criteria(s): The user should be ableto record more data for themselves.</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.</td></tr>
<tr><td colspan="3">Steps:

• Run the Capture_Signs.ipynb
• Type a label that doesn't exist in the data set
• Type a number of sequences that's the same as the rest ofthe data.
• Press Start Button.
• Record all sequences.</td></tr>
<tr><td>Expected: New data should be stored in the Raw_data file. This datawill need to be run through the model though</td><td>Actual:
A folder for the newdata appeared with the correct number of sequences all filled with frame data.</td><td>PASS</td></tr>
</table>

Table 8: Test Case 6

<table>
<tr><td colspan="2" rowspan="2">Test Number: 7</td><td>Aim: The application runs fast and smoothly</td></tr>
<tr><td>Acceptance Criteria(s): The video feed of the user is not slowed down by the prediction processing</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.
• Point the webcam at your self to show your torso and face.
• Open the application
• move hands below the screen before application opens.</td></tr>
<tr><td colspan="3">Steps:

• Move hand up into a waving position
• Wave continually
• Monitor the speed at which the frame and wait counter increase.
• Monitor the visible frame speed of yourself in the camera
• Observe for any jolts in camera performance near frame 15 / early wait.</td></tr>
<tr><td>Expected: The camera should not stutter or lag when a prediction is made. The user should see the same frame rate from the camera as they did throughout.</td><td>Actual:
The frame rate although not high speed does not change when the prediction is made.</td><td>PASS</td></tr>
</table>

Table 9: Test Case 7

<table>
<tr><td colspan="2" rowspan="2">Test Number: 8</td><td>Aim: The application runs fast and smoothly</td></tr>
<tr><td>Acceptance Criteria(s): The application can present a prediction to the user before they move to their next sign (keeping it real time).</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Set up an external webcam.
• Point the webcam at your self to show your torso and face.
• Open the application
• move hands below the screen before application opens.</td></tr>
<tr><td colspan="3">Steps:

• Perform Test1 Steps
• Monitor time in which sign translation appears after frame hits 15/15 .</td></tr>
<tr><td>Expected: The application should output the predicted sign before wait reaches 10/10</td><td>Actual:
The sign translation appeared instantly after the frame reached 15.</td><td>PASS</td></tr>
</table>

Table 10: Test Case 8

<table>
<tr><td colspan="2">Test Number: 9(Note; May take long to copy files, and worth reducing epochs.)</td><td>Aim: The application runs fast and smoothly
Acceptance Criteria(s): The application Doesn't slow down as the data set grows..</td></tr>
<tr><td colspan="3">Pre-Conditions:

• Perform test 7 8
• Observe the speed of the application.</td></tr>
<tr><td colspan="3">Steps:

• Copy and paste the contents of the Raw_Data folder, into it'self.
• Rename the items from x – copy to a random words
• Re-train the model.
• Run the translation application.
• Observe the times.

  – Note that it's likely the translations are wrong.</td></tr>
<tr><td>Expected: The application should run at roughly the same observedspeed as before.</td><td>Actual:
The application seems to run the same.</td><td>PASS</td></tr>
</table>

Table 11: Test Case 9

IX Future Work

Overall, I believe I have managed to make an application that meets the aims set out at the start of this project. My product can translate signs from the user's camera feed in real time. The product demonstrates a small selection of BSL signs but has the functionality to be trained with more signs, and signs from other localisation's. In this segment I will talk about some of the future work I would like to carry out to make this product better.

IX.I Code Work

IX.I.I Code-Rewrite

Before continuing any work on the project, I would take a good amount of time to refactor most of the code. I believe as I have worked on the project, I have learnt a lot more about the packages and libraries I have used and would be able to use that knowledge to better write the program again. Rewriting the program as the first step of future work would give a better base product in which to enhance and further with the other future work. Overall, the products code is easy to understand and well documented but using new classes and separating some segments of code out further would increase readability and decrease complexity.

Introducing threading to the program would help increase the speed in which frames could be displayed to the user as predictions could be done on a separate thread. Threading is complicated and would benefit from being introduced during the re-write rather than after.

IX.I.II Error Handling

Currently the products that I have created have little to no error handling. This means when the user encounters an error the product just crashes, closes unexpectedly, or shows a complicated error from the source code. Both the

user and the developer would benefit from error handling. For the developer, logs could be sent to them whenever a user's application crashes, breaks, or performs unexpectedly. This would help them make the application better with bug fixes and announcements to the community when they discover problems. The user would benefit as they would have abetter understanding as to what went wrong with their program, they may be able to trouble shoot the issue themselves if enough information is given.

IX.I.III User Interface Redesign

While the product I have developed does currently have an interface that displays relevant and important information to the user, it does not look very nice. PySimpleGUI is exactly what it says, a simple graphical user interface. It would be better redesigned with a more complex but better-looking package that would make the product look enterprise worthy. Designing the application with UX in mind would allow users to have a better experience on the product as they would be more likely to understand the flow and meaning of the options and sections available to them. As I will speak about later, where another option is moving the application to mobile, the product could be designed in a web architect instead.

IX.I.IV API

In the future I would like to add an API that might allow other applications to use our translation service. Take for example a company like Facebook want to work with us. They could have a toggle on their video calling service for translations, that would send to our server their sequence data and we would return our predicted translation. Opening our work to other companies and software providers would be beneficial as the whole point of this project is about accessibility, and what better way than making sure our work can be used by other to further benefit the community.

IX.II New Features

IX.II.I Mobile

Porting the product onto other devices would be useful. Currently we're limited to using it on a windows machine. There are two main options I would like to explore. One option would be moving the application to the web. This means we could have it hosted on a website and have users connect from any device they have. The other option would be to slightly limit the scope to mobile applications. This could be both android and apple and would allow people to make translations directly from their phone camera. Since most modern mobile phones have cameras, and most people have a modern mobile phone it would be reasonable to assume a mobile phone to be the most likely used device for our product.

IX.II.II Exports

Another feature that would be good to add would be better exporting functionality. Currently the user can only copy and paste their translation out of the application but integrating with other communication providers such as Facebook Messenger, WhatsApp may allow people to share their translations more easily to the people they're for. This could be implemented relatively easily as it would just require people to connect their accounts a few calls to the API provided by these companies for sending messages. Providing more ways in which the resulting output could be used would make the service more relevant. Furthering this idea would be the actual creation and implementation of a calling service using this technology to directly allow people to video call another with their sign language translations.

IX.II.III Fluent Signing

Fluent signing would be one of the best things to implement into the product. Currently were limited to performing a single sign in each prediction

window. This was a limitation of the time I had available to complete this project and I struggled to find a way around this. When experimenting with continuous prediction, the hand movements between sign starting positions often confused the prediction model into bad predictions. Fluent signing could be achieved by adding compound signs, such as "Hello my name is" instead of "hello", "my", "name", "is". Alternatively, more research would be needed to find a better way for it to predict these longer strings. Fluent signing could be better reinforced by a larger data set for each sign.

IX.III Community

IX.III.I Switching Data

Since the translator app only depends on the model and label map, it's possible to just switch them out for alternative files. In its current state this would have to be done manually and placed in the correct locations. I believe, with more reasoning in the next point that it would be a beneficial endeavour to allow some automation for switching models. Switching models easily would allow for different localisation's to be loaded in quickly. This means you could go from translating ASL to BSL in a flash. A library of different models could be bundled by default with the application to help support a wider audience.

IX.III.II Crowd-sourcing Community

The final piece of future work that I would like to do to this project is to make it more crowd sourced and allow others to contribute. I think this would be a great platform for users to upload their own data sets and contribute to others. This platform could help create some of the biggest data sets available of sign language, which could in turn be used not only by our application but also other companies looking to make accessibility software. Our product would better evolve as more people contribute and other people's

results would be more reliable meaning both parties would win. Currently the model I have is good, but it takes time to manually gather the training data as a single person. This solves this problem in an ingenious way.

X Conslusion

The project has delivered a set of three applications that a user can use for translating sign language to text, and enhancing the provided model with new signs or more data. The project had three main aims which it needed to meet in order to be successful. The three aims were that When the user performed a sign, a text translation was provided, that the application was easily adaptable to incorporate more signs and gestures and, that the application ran fast and smoothly. Through prolonged research into relevant techniques and the user of useful packages I can happily say that these aims have been met. The application in it's current state demonstrates the ability for sign language to be translated to text. The application offers tools that would allow the usefulness of the application to be grown through the introduction of new and extended data sets.

The application was tested to measure it's accuracy. Initial reporting by the model showed a high degree of accuracy. When tested by the user the model performed slightly worse but still to a reasonably high level of accuracy, demonstrating its overall effectiveness. I explored the reasons behind the discrepancies in accuracy as well as exploring other ways in which the program could later be enhanced. Having a limited scope meant that many exciting features did not make it into the program. Enhancing the current solution with these additions would give a more useful product. As part of this dissertation I have also reflected on the learning achieved through it's undertaking. This has allowed me to reflect on the new knowledge I have gained and how I would perform this project differently in the future.

From the research I carried out on this project it's clear there is never an exact path for machine learning. I believe applications like this may exist properly in the near future as technology continues to enhance. many papers I read talked about their attempt at making a better version of the RNN, CNN, GRU, or LSTM model. Research like this will quickly open up developers to better, more efficient learning algorithms which will help

production of this and similarly needing machine learning tasks.

BSL and sign language in general is an interesting language and I'm happy to to have had the experience to gain a better understanding of it. The difficulties for deaf and hearing impaired individuals is something most of us cannot imagine. I'm pleased to have worked on a project that explores ways in which to make the world more accessible for them.

XI Reflection On Learning

Quite a lot of the focus of the MSc dissertation is research. During the initial phase I spent a good deal of time researching different methods that I may use to implement my project. This research was invaluable, allowing me to save time and effort by focusing my time effectively. I believe researching takes time, not just to find the items you're looking for but to also think about the ideas you find and process them into your own solutions. I'd like to positively identify how research helped my topic form in the way it did, with the addition of items like MediaPipe which I previously would not have used. Research also came in the form of R&D, where I tested python packages and scripts on different branches to see how they could improve my project.

One major topic that I got to expand my knowledge on this project was machine learning. Having previously only completed projects that implemented CNN's I was able to further my knowledge by researching and using RNN's in my work. Broadening my knowledge in the machine learning field through a good mixture of research and personal implementation means I'm much better equipped to carry out a machine learning task in the future. Before this year I had not previously used machine learning and finishing my masters with a handful of projects that incorporate machine learning is great.

Completing this project on my own has allowed me to increase and demonstrate my project development skills once again greatly. Source control software such as Git is used by almost every large technology company. Having the opportunity to brush up on my commands and source control practices in a real project scenario has been ideal. Practicing and refreshing my knowledge on other development practices like Agile and Kanban have also been good. Moving into a software development job in the next few weeks, this project has allowed me to get back up to speed with these processes I will likely be using very soon. Managing a project for myself has shown me the

importance of documentation and procedure as it has enforced good habits and coding practices. Moving forward If I was to work on a project again soon, I would like to make more use of development boards like Trello. Although it was an item, I used to help me document and break up the work I had, I did not utilise it fully and just used it as a template. Implementing sprints alongside development boards may ensure my work is of a better quality – with items like sprint reviews improving performance over coming weeks.

Working with sign language has been a great learning experience. I have had the opportunity to spend time learning about a language that is used by many people here in the UK. Seeing firsthand how simple some of the signs can be to do and remember has enlightened me to make some effort outside of this project to further learn the language. After this project I believe it should be a skill that everyone tries to do or is taught to children at a younger age.

I would lastly like to reflect on how finding the right project helps your own learning. It's important to be interested in the topic you're doing. This often means, choosing your own project will help give you a better learning outcome. My supervisor told me he believed that choosing your own topic gives you more passion about what you're doing. This is a statement I agree with fully. From the getgo of personally choosing my topic I was excited to see something come to fruition and to create a solution the problem I had identified.

Bibliography

[1] British Deaf Association. *WHAT IS BSL?* URL: https://bda.org.uk/help-resources/ (visited on 08/01/2021).

[2] Google. *Live ML anywhere*. URL: https://mediapipe.dev/ (visited on 08/08/2021).

[3] nidcd.nih.gov. *American Sign Language*. URL: https://www.nidcd.nih.gov/health/american-sign-language (visited on 08/01/2021).

[4] www.atlassian.com. *What is Agile?* URL: https://www.atlassian.com/agile (visited on 08/05/2021).

[5] www.british-sign.co.uk. *Sign Language*. URL: https://www.british-sign.co.uk/ (visited on 08/03/2021).

[6] www.british-sign.co.uk. *What is British Sign Language?* URL: https://www.british-sign.co.uk/what-is-british-sign-language/ (visited on 08/03/2021).

[7] Ph.D. Daphne Bavelier, Elissa L. Newport, and Ph.D. Ph.D.and Ted Supalla. *Children Need Natural Languages, Signed or Spoken*. URL: https://dana.org/article/children-need-natural-languages-signed-or-spoken/ (visited on 08/03/2021).

[8] K. Emmorey and H.L. Lane. *The Signs of Language Revisited: An Anthology To Honor Ursula Bellugi and Edward Klima*. Taylor & Francis, 2013. ISBN: 9781135669003. URL: https://books.google.co.uk/books?id=0MoXjMX3p8UC.

[9] www.wfdeaf.org. *our work*. URL: https://wfdeaf.org/our-work/ (visited on 08/03/2021).

[10] www.census.gov. *U.S. and World Population Clock*. URL: https://www.census.gov/popclock/ (visited on 08/04/2021).

[11] www.ons.gov.uk. *Population estimates*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates (visited on 08/04/2021).

[12] rnid.org.uk. *Facts and figures*. URL: https://rnid.org.uk/about-us/research-and-policy/facts-and-figures/ (visited on 08/04/2021).

[13] Cyrus John. *Demand For Video Conferencing Apps Zoom During COVID-19 Lockdown*. URL: https://www.thequint.com/tech-and-auto/tech-news/growth-in-video-chat-apps-download-during-lockdown#read-more (visited on 08/04/2021).

[14] rnid.org.uk. *How video conferencing apps compare for accessibility*. URL: https://rnid.org.uk/2020/12/how-video-conferencing-apps-compare-for-accessibility/ (visited on 08/04/2021).

[15] Loris Bazzani et al. "Self-taught object localization with deep networks". In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Mar. 2016). DOI: 10.1109/wacv.2016.7477688.

[16] Joos Korstanje. *What is the difference between Object Detection and Image Segmentation?* URL: https://towardsdatascience.com/what-is-the-difference-between-object-detection-and-image-segmentation-ee746a935cc1 (visited on 08/09/2021).

[17] Anagha Dhote and S. C. Badwaik. "Hand tracking and gesture recognition". In: *2015 International Conference on Pervasive Computing (ICPC)* (Apr. 2015). DOI: 10.1109/pervasive.2015.7087108.

[18] Google. *MediaPipe Holistic*. URL: https://google.github.io/mediapipe/solutions/holistic (visited on 08/08/2021).

[19] Ivan Grishchenko and Valentin Bazarevsky. *MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device*. URL: https://google.github.io/mediapipe/solutions/holistic.html (visited on 08/08/2021).

[20] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. "Differential Recurrent Neural Networks for Action Recognition". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4041–4049. DOI: 10.1109/ICCV.2015.460.

[21] Akram Mihanpour, Mohammad Javad Rashti, and Seyed Enayatallah Alavi. "Human Action Recognition in Video Using DB-LSTM and ResNet". In: *2020 6th International Conference on Web Research (ICWR)*. 2020, pp. 133–138. DOI: 10.1109/ICWR49608.2020.9122304.

[22] Behrooz Mamandipoor et al. "Monitoring and detecting faults in wastewater treatment plants using deep learning". In: *Environmental Monitoring and Assessment* 192 (Feb. 2020). DOI: 10.1007/s10661-020-8064-1.

[23] www.tensorflow.org. *An end-to-end open source machine learning platform*. URL: https://www.tensorflow.org/ (visited on 08/25/2021).

[24] www.tensorflow.org. *The Sequential model*. URL: https://www.tensorflow.org/guide/keras/sequential_model (visited on 08/25/2021).

[25] www.tensorflow.org. *Recurrent Neural Networks (RNN) with Keras*. URL: https://www.tensorflow.org/guide/keras/rnn (visited on 08/25/2021).

[26] www.tensorflow.org. *tf.keras.layers.LSTM*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM (visited on 08/25/2021).

[27] pytorch.org. *FROM RESEARCH TO PRODUCTION*. URL: https://pytorch.org/ (visited on 08/26/2021).

[28] pytorch.org. *SEQUENTIAL*. URL: https://www.tensorflow.org/guide/keras/sequential_model (visited on 08/26/2021).

[29] pytorch.org. *LSTM*. URL: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html?highlight=lstm (visited on 08/26/2021).

[30] Mihai Cristian Chirodea et al. "Comparison of Tensorflow and PyTorch in Convolutional Neural Network - based Applications". In: *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2021, pp. 1–6. DOI: 10.1109/ECAI52376.2021.9515098.

[31] google.com. *Apps*. URL: https://play.google.com/store/search?q=sign%20language%20translation (visited on 09/19/2021).

[32] https://economictimes.indiatimes.com/. *Meet the new Google translator: An AI app that converts sign language into text, speech*. URL: https://economictimes.indiatimes.com/magazines/panache/meet-the-new-google-translator-an-ai-app-that-converts-sign-language-into-text-speech/articleshow/66379450.cms (visited on 09/19/2021).

[33] nicknochnack. *ActionDetectionforSignLanguage*. URL: https://github.com/nicknochnack/ActionDetectionforSignLanguage (visited on 09/19/2021).

[34] Tao Liu, Wengang Zhou, and Houqiang Li. "Sign language recognition with long short-term memory". In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 2871–2875. DOI: 10.1109/ICIP.2016.7532884.

[35] Anusorn Chaikaew, Kritsana Somkuan, and Thidalak Yuyen. "Thai Sign Language Recognition: an Application of Deep Neural Network". In: *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Elec-*

tronics, Computer and Telecommunication Engineering. 2021, pp. 128–131. DOI: 10.1109/ECTIDAMTNCON51128.2021.9425711.

[36] opencv.org. *Home - OpenCV*. URL: https://opencv.org/ (visited on 08/13/2021).

[37] opencv.org. *OpenCV modules*. URL: https://docs.opencv.org/master/ (visited on 08/13/2021).

[38] Yea-Shuan Huang, Yu-Chung Chen, and Fang-Hsuan Cheng. "Hand gesture detection and extraction". In: *2013 IEEE China Summit and International Conference on Signal and Information Processing*. 2013, pp. 669–673. DOI: 10.1109/ChinaSIP.2013.6625426.

[39] M. P. Paulraj et al. "Extraction of head and hand gesture features for recognition of sign language". In: *2008 International Conference on Electronic Design*. 2008, pp. 1–6. DOI: 10.1109/ICED.2008.4786633.

[40] Wen-Pinn Fang. "An Intelligent Hand Gesture Extraction and Recognition System for Home Care Application". In: *2012 Sixth International Conference on Genetic and Evolutionary Computing*. 2012, pp. 457–459. DOI: 10.1109/ICGEC.2012.57.

[41] Camillo Lugaresi et al. "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. URL: https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf.

[42] Thamindu DJ. *Crop and Select Only the Detected Region from an Image in Python*. URL: https://stackoverflow.com/questions/55204998/crop-and-select-only-the-detected-region-from-an-image-in-python (visited on 10/07/2021).

[43] www.tutorialspoint.com. *TensorFlow - CNN And RNN Difference*. URL: https://www.tutorialspoint.com/tensorflow/tensorflow_cnn_and_rnn_difference.htm (visited on 08/13/2021).

[44] www.atlassian.com. *What is kanban?* URL: https://www.atlassian.com/agile/kanban (visited on 08/13/2021).

[45] trello.com. *Trello*. URL: https://trello.com/en-GB (visited on 08/13/2021).

[46] git-scm.com. *git*. URL: https://git-scm.com/ (visited on 08/13/2021).

[47] github.com. *github*. URL: https://github.com/ (visited on 08/13/2021).

[48] tortoisegit.org. *The Power of Git in a Windows Shell*. URL: https://tortoisegit.org/ (visited on 08/13/2021).

[49] pysimplegui.readthedocs.io. *Python GUIs for Humans*. URL: https://pysimplegui.readthedocs.io/en/latest/ (visited on 08/14/2021).

[50] Levente Pető and János Botzheim. "Parameter Optimization of Deep Learning Models by Evolutionary Algorithms". In: *2019 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*. 2019, pp. 000027–000032. DOI: 10.1109/IWOBI47054.2019.9114508.

[51] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (Feb. 2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199.

[52] jupyter.org. *jupyter*. URL: https://jupyter.org/ (visited on 08/01/2021).

[53] python.org. *Miscellaneous operating system interfaces*. URL: https://docs.python.org/3/library/os.html#os.listdir (visited on 08/27/2021).