A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science

in

Big Data & Business Intelligence

Real-Time British Sign Language Recognition System

Name: Dimpalkumari Panchal

Student ID: 001323903

Supervisor: Dr. Hai Huang

Submission Date: 06 September 2024

Word Count: 12854

REAL-TIME BRITISH SIGN LANGUAGE RECOGNITION SYSTEM

XXXXX
Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

(Submitted 6 September 2024)

ABSTRACT: The aim of this project is to design and implement a real-time British Sign Language (BSL) recognition system using deep learning approaches. I aim at translating utterances from BSL to English employing a Gated Recurrent Unit (GRU) model as our base network, which has been shown to produce similar performance than popular models like Long Short-Term Memory albeit with much less computational needs. Using hand, face and body pose landmarks, the system identifies 30 pre-defined BSL gestures from images. The keypoints were detected using MediaPipe and OpenCV on the collected data before it was used to train models. While these challenges still persist, the final GRU model obtained an accuracy of 90%. This work could be improved by optimizing latency, increasing the dataset for training and evaluating performance in varied environment. This study thus presents a promising application of deep learning in real-time sign language recognition, which will lead to further improvement on inclusivity for our communication technology.

Keywords: Sign Language Recognition, British Sign Language, Gated Recurrent Units, Real-Time Gestures Recognition, Deep Learning, OpenCV, Mediapipe.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Hai Huang, for his invaluable guidance, support, and encouragement throughout the course of this project. I would like to thank both Dr. Hai Huang and Sanyaade Olufemi Adekoya for agreeing to conduct the project demonstration on the scheduled day.

I want to express my sincere gratitude to my flatmate, Ankit Raj, and my peer Sakshi Pardhi, for their unwavering help and support. Their assistance in testing real-time detection was crucial to this project's conclusion. Lastly, I want to express my gratitude to family for their continuous encouragement and support during my academic career.

Table of Contents

Abstract ..... ***Error! Bookmark not defined.***

Acknowledgements ..... *ii*

Table of Contents ..... ***Error! Bookmark not defined.***

List of Tables ..... *vii*

List of Figures ..... *viii*

1. Introduction *1*

1.1. Overview *1*

1.2. Problem Statement *2*

1.3. Research Objectives *2*

1.4. Scope of the Project *3*

1.5. Methodology Overview *3*

1.6. Road Map of Report *4*

2. Literature Review *6*

2.1. Overview of Sign Language Recognition *6*

2.2. Machine Learning Approaches in Gesture Recognition *6*

2.2.1. Hidden Markov Models (HMM) *6*

2.2.2. Support Vector Machines (SVM) *7*

2.2.3. Neural Networks (NN) *7*

2.3. Deep Learning Techniques for Gesture Recognition *8*

2.3.1. Convolutional Neural Networks (CNN) *8*

2.3.2. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) *8*

2.3.3. Gated Recurrent Units (GRU) *9*

2.4. Datasets in Gesture Recognition *9*

2.5. Real-Time Sign Language Recognition *10*

2.6. Summary *11*

3. Background Research 12

3.1. Introduction 12

3.2. Primary Research 12

3.2.1. Data Collection 12

3.2.2. Data Preprocessing 12

3.3. Secondary Research 12

3.3.1. Sign Language Recognition 12

3.3.2. Advances in Deep Learning for Gesture Recognition 13

3.3.3. British Sign Language: Unique Challenges and Considerations: 14

3.4. Summary 14

4. Legal, Social, Ethical and Professional Issues 15

4.1. Introduction 15

4.2. Legal Issues 15

4.3. Social Issues 16

4.4. Ethical Issues 16

4.5. Professional Issues 17

4.6. Summary 18

5. Methodology 19

5.1. Introduction 19

5.2. Project Set-up and Environment Configuration (Installing Dependencies & Project Structure) 19

5.3. Data Collection of BSL Gestures 20

5.3.1. Folder Set-Up 20

5.3.2. Function Creation 20

5.3.3. Data Collection 22

5.3.4. Challenges and Considerations 24

5.4. Preprocessing *24*

5.5. Model Development *24*

5.5.1. Exploration of Various Models *24*

5.5.2. GRU Model Optimization *26*

5.6. Implementation *26*

5.6.1. Integration with OpenCV for Real-Time Gesture Recognition *26*

5.6.2. Custom Visualisation with MediaPipe *27*

5.7. Evaluation & Testing *27*

5.7.1. Evaluation Metrics *27*

5.7.2. Testing the Performance under Various Conditions *27*

5.8. Summary *28*

6. Development of the Model *29*

6.1. Introduction *29*

6.2. System Optimization and Final Model *29*

6.2.1. Hyperparameter Tuning *29*

6.2.2. Optimization with GridSearchCV *33*

6.2.2.1. Initial Grid Search and Best Parameters *33*

6.2.2.2. Expanded Grid Search *33*

6.2.2.3. Challenges with Expanded Grid Search *34*

6.2.2.4. Strategic Decision to Revert to Original Configuration *34*

6.3. Summary *34*

7. Implementation and Testing *35*

7.1. Final Model Architecture *35*

7.2. Real-Time BSL Detection *36*

7.3. Summary *38*

8. Evaluation and Analysis *39*

8.1. Introduction *39*

8.2. Results and Visualisations of GRU Model *39*

8.3. Critical Review *41*

8.4. Challenges and Observations *44*

8.5. Comparison and Discussion *45*

8.6. Summary *46*

9. Conclusion and Future Work *47*

9.1. Conclusion *47*

9.2. Future Work *47*

9.3. Final Thoughts *49*

References *50*

List of Tables

Table 1: Results of Various Hyperparameter Values 32
Table 2: Result Comparison of Various Models 45

List of Figures

Figure 1: Structure of Data Files 20
Figure 2: British Sign Language Gestures (British Greeting Signs) 22
Figure 3: Recorded Gestures for Data Collection 23
Figure 4: Recording Gestures for Data Collection(2) 23
Figure 5: Accuracy and Loss Curve 40
Figure 6: Confusion Matrix 41
Figure 7: Real-Time Sign Language Recognition with Different User (1) 43
Figure 8: Real-Time Sign Language Recognition with Different User (2) 43

1. Introduction

1.1. Overview

For the hearing-impaired and speech impaired, Sign Language is an important means of communication. Without it, they may find difficulty in interacting with the world. However, there remains a significant gap in the seamless communication between people who use sign language and those unfamiliar with it. Many individuals diagnosed with Amyotrophic Lateral Sclerosis and Parkinson's disease suffer from speech problems as their symptoms include weakening and stiffing of muscles in the body (Felman,2024). They have slurred or no functional speech at all, with weak muscles in the arms and legs preventing them from using mobile devices to communicate by tying or even writing on paper. Inequality like this is what makes sign language recognition technologies so important, responsive, and accessible to address these communication challenges. The recent developments in the field of machine learning and computer vision have led towards an automated system for sign language recognition. Most of these systems have the ability to identify sign language gestures and translate them into spoken or written English, allowing a much broader spectrum of communication.

The objective of this project is to create a precise and effective system for sign language recognition which can provide instant translations using advanced deep learning methods, with a specific emphasis on the British Sign Language (BSL). This project investigates the possibility of Gated Recurrent Units (GRUs) instead of commonly used Long Short-Term Memory (LSTM) or Convolutional Neural Network (CNN) models because studies claim that GRUs can provide a similar performance while being less computationally expensive. People recognise Gated Recurrent Units (GRU's), a type of Long Short-Term Memory (LSTM) variation, for their simplified structure and rapid training speed. The system is specifically developed to identify and interpret gestures related to British Sign Language (BSL), converting them into written text as and when they are being performed. This allows individuals who do not understand sign language to access and comprehend the communication. This chapter provides an overview of the project's background, goals, approach, and the organisation of the report.

1.2. Problem Statement

This language barrier between people who use sign language and those who don't, and the inability to communicate for patients facing speech impairments along with muscle weaking is one of the main problems. Traditional methods aimed at closing this gap, such as in-person interpreters, can be expensive or hard-to-diffuse logistically. It is not possible to have an in-person interpreter present at all public spaces. And in today's fast-growing world, everyone likes to live independently. This leaves a gap for automated systems that can accurately interpret sign language in real-time (Alyami *et al*., 2024). Gestures are not very easy to identify; they differ heavily in motion speed and context. As a result, developing those sorts of systems is difficult to do. Furthermore, existing systems often fail to deliver both accuracy as well as real-time performance, making them infeasible for practical deployment.

The goal of this project is to experiment on the success ratio of Gated Recurrent Units (GRUs) for recognising sign language, as many other attempts have used Long Short-Term Memory (LSTM) or Convolutional Neural Network (CNN) models in prior research and development. The goal is to develop an accurate Sign Language Recognition System using a GRU model that can appropriately identify a defined set of British Sign Language (BSL) gestures when performed in front of a camera and display the translated test in the form of the output. The project strives for high precision and real-time execution efficiency with every-day usability in mind.

1.3. Research Objectives

The primary objectives of this project are:

- To develop a deep learning model that can identify and comprehend a specific set of pre-defined British Sign Language (BSL) gestures with accuracy.
- To evaluate the performance of the Gated Recurrent Unit (GRU) model for gesture recognition.
- To increase the accuracy of the Gated Recurrent Unit (GRU) model with hyperparameter tuning and other optimisation techniques to measure highest precision.
- To develop a system for instantaneously recognising sign language and one which can be employed in practical applications, e.g., as part of mobile apps or assistive devices;

- To analyse and evaluate the accuracy, precision sensitivity, and performance under various conditions of the existing system, along with guidelines for future tests with variety in near-to-actual setup.
- To compare the GRU model's accuracy with the more commonly used models like LSTMs and CNNs.

1.4. Scope of the Project

This project is aimed at visualising and making instant predictions of a distinct set of 30 gestures in British Sign Language (BSL). These gestures were manually selected to form an extensive set of commonly used signs, making the system practical as most people use these signs daily.

The project begins with the collection of gesture data, preprocessing, feature extraction, model training, and real-time testing. The structure of this project stands out from most existing systems that only consider hand gestures, as it combines face and position landmarks to enhance the accuracy and robustness of the recognition system. The system uses a holistic perspective, allowing it to understand more complex gestures than simple hand movements, including facial expressions and body position.

This project aims to achieve a high level of accuracy in predicting a group of gestures that are suitable for real time performance. This is an idea in principle and will be useful as a model whose vocabulary can be expanded. The processes developed in this project are designed to be flexible for future extensions containing additional signs and other sign languages.

1.5. Methodology Overview

The methodology used for this project comprises various essential stages:

- **<u>Data Collection and Preprocessing</u>**

  The project started by collecting 30 sequences per gesture of hand, face, and position movements with the use of OpenCV. Each sequence consisted of 30 frames. The data was preprocessed to extract important features such as key points from each gesture, and map each of them to a numerical label. The features were then used to train the recognition model.

- **Model Development and Training**

  Various deep learning models, including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network-based models, were experimented with. The GRU model had promising performance, and hence, it was selected for further improvement to explore whether it could provide a similar performance to the more commonly used LSTM based models. Hyperparameters of the model were tuned manually and with the help of GridSearchCV to improve performance. which resulted in a robust recognition system.

- **Real-Time Implementation**

  The final model was deployed as a real-time system capturing live video feed and processing each frame to extract key points. It used the trained model to identify gestures and convert them into text.

- **Evaluation and Testing**

  The accuracy, precision, recall, and real-time performance of the recognition system were evaluated. This included a thorough examination of the model's strengths, limitations, and areas for improvement. The performance was compared with that of an already experimented LSTM and CNN model.

1.6. Road Map of Report

This report is organised into various chapters, with each one focussing on a crucial aspect of the project.

- Chapter 1: Introduction

  It gives a broad outline of the project, beginning with an overview, the problem statement, research objectives, the scope of the project, methodology, and the report structure.
- Chapter 2: Literature Review

  It discusses previous studies conducted in the area of sign language recognition, machine learning techniques used, and related datasets useful for training such models.
- Chapter 3: Background Research

It focusses on the primary and secondary research conducted for this project.

- Chapter 4: Legal, Social, Ethical and Profession Issues
  It outlines the legal considerations, social implications, ethical issues and professional concerns of this project.
- Chapter 5: Methodology
  It explains the workflow of the project consisting of data collection, preprocessing, model development, GRU model optimization and implementation.
- Chapter 6: Development of the System
  It describes the design, training, and optimization methods to build the model in detail.
- Chapter 7: Implementation and Real-Time Testing
  It discusses how the model was ultimately executed in a real-time sign language recognition system, as well as tested for performance under realistic circumstances with various users.
- Chapter 8: Evaluation and Analysis
  It evaluates the final model, compares its performance with existing models, and discusses its strengths and weaknesses while making recommendations for further improvements.
- Chapter 9: Conclusions and Future Work
  It shows a consolidation of the work with a focus on results, comparison to research objectives, and suggestions for future work.

2. Literature Review

2.1. Overview of Sign Language Recognition

State-of-the-art sign language recognition systems have developed enormously in recent times, largely due to advances in the fields of machine learning and computer vision. First-wave systems mainly used the most advanced method at this time, handcraft representation with traditional machine learning algorithms that made these methods limited in generalisability over different users and environmental conditions. In the beginning, some of these systems would differentiate hand postures and gestures (occasionally even facial expressions) that converted directly into written or vocal text. But these approaches struggled with the variability present in sign language from variations of facial expressions, signing styles, speeds, and contextual meanings.

However, deep learning with Convolutional or Recurrent Neural Networks has been a game changer in the design of sign language recognition systems. Such models have exhibited an impressive capacity to perform automatic feature learning from raw data, resulting in dramatically increased recognition accuracy and robustness. However, some challenges are yet to be solved effectively, such as achieving real-time performance and working with a large diversity of gestures while keeping the accuracy high in different real-world scenarios.

2.2. Machine Learning Approaches in Gesture Recognition

2.2.1. Hidden Markov Models (HMM)

To model the sequential data inherent in sign language, Hidden Markov Models (HMM) were extensively used in early sign language recognition systems. HMMs use the sign language gesture as sequences of states (i.e., each state corresponds to some sort of hand shape or movement). The transitions between states are also probabilistic, learnt from the training data. In fact, HMMs are well suited to represent the temporal dynamics of sign language and are good at modelling continuous sing languages where one gesture flows into another.

While HMM is simple to use, its power can be limited by the need for

previously defined states and transitions, which may constrain appropriately modelling each sign language gesture in an expressive way. As concluded in their paper (Assan and Grobel, 1998), HMMs require a large amount of labelled training data to accurately estimate the probable transitions, which makes them limited in practical applications.

2.2.2. Support Vector Machines (SVM)

Support Vector Machines (SVMs) have been used in sign language recognition systems, especially for gesture classification. SVMs are a type of high-performance classifier, that find an optimal hyperplane that can separate the data into classes with maximum margin. Therefore, SVMs are a great option for handling high-dimensional data like the extracted features from sign language gestures.

SVMs have been applied in the field of sign language recognition to classify hand shapes or movements based on features extracted through images or video frames. However, SVMs do not work very well when it comes to more complex tasks in sign language with overlapping classes or where the number of classes increases. Moreover, SVMs do not perform as well for dynamic and continuous sign language recognition (Chong and Lee, 2018).

2.2.3. Neural Networks (NN)

Neural Networks, when first introduced, were a milestone for sign language recognition. The early neural networks that made inroads on intermediate processing were mostly feedforward of some variation or another, and these gave something near to state-of-the-art accuracy compared with other traditional models like Hidden Markov Models (HMMs) and Support Vector Machines (SVMs). These networks had the ability to learn more sophisticated features from data and hence could recognise sign language signs better.

However, when sign language entered the game with all its time-based aspects, it became clear that feedforward nets were very limiting. Sign language gestures are sequential, their meaning depends on their order and timing of the gesture. Traditional feedforward networks, which act on each input

independently, were insufficient to capture these temporal dependencies. This motivated researchers to design Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs) (Zheng, Liang and Liang, 2017).

2.3. Deep Learning Techniques for Gesture Recognition

2.3.1. Convolutional Neural Networks (CNN)

The most widely adopted structure for gesture recognition is the Convolutional Neural Network (CNN) because it can automatically extract spatial features from images and video frames. Convolutional Neural Networks (CNNs) accomplish this by applying a set of convolution filters to the input data, which capture patterns like edges, textures, and shapes that are fundamental in recognising hand gestures.

For sign language recognition, CNNs are typically applied to each frame in a video sequence to extract features such as hand shape, orientation, and position. These features are then passed on as input to subsequent layers or other models (e.g., RNN/ LSTM), which handle the temporal dynamics of this sequence of gestures.

It comes with a few limitations, the most significant of which is that while CNNs intend to capture spatial information well, they are not optimised for capturing temporal dependencies. As discussed by Ma, Zu and Kim (2022) to overcome this limitation, conventional CNNs are usually in combination with recurrent architectures such as LSTMs or GRUs where the temporal sequence of gestures can be learnt.

2.3.2. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) were one of the earliest deep learning architectures that was built explicitly to work on sequential data and for applications such as sign language recognition. RNNs keep a hidden state that is modified at every time step; this way, the network can store information from previous inputs and make predictions across an entire sequence.

Standard RNNs, however, have the problem of vanishing and exploding gradients, which may cause difficulty in learning long-term dependencies. Long Short-Term Memory (LSTM) networks were designed to address these issues. LSTMs come with memory cells and gating mechanisms that let the network pay attention to or forget specific information only if necessary — which is precisely what makes it so good at capturing those long-term dependencies.

Especially for sign language recognition, various tasks are based on understanding of the temporal sequences of gestures, which has made LSTMs a very efficient option. Mienye, Swart and Obaido (2024) say that they can increase recognition accuracy, especially when combined with CNNs to extract features. But LSTMs can be computationally expensive, and they need a lot of data to train well.

2.3.3. Gated Recurrent Units (GRU)

Gated Recurrent Units (GRUs) are a more recent development in the deep learning universe that introduced us to another streamlined and computationally efficient replacement of LSTMs. Since GRUs use a single update gate instead of separate forget and input gates (LSTMs), they have fewer parameters to train, hence the training time is faster.

Despite being so simple, in several tasks, including sign language recognition, GRUs have been shown to outperform LSTMs. GRUs are a good choice for real-time applications as they have lower computational requirements and train faster (Chakraborty, S., et al., 2023). However, the use of GRUs in sign language recognition is less considered with respect to common choices, which are LSTMs. This creates the possibility for further research and experimentation.

2.4. Datasets in Gesture Recognition

Effective sign language recognition systems rely on datasets that are both available and high quality. Over the years, many datasets have been created to aid research in

this area; from small manually collated datasets to large publicly available ones. The most used datasets are:

- RWTH-BOSTON-104: A common dataset for continuous sign language recognition, offers over 104 American Sign Language (ASL) sentences.
- ASLLVD (American Sign Language Lexicon Video Dataset): A huge collection of thousands of American Sign Language (ASL) signs depicted through videotaped entries from multiple signers in a variety of viewpoints.
- PHOENIX-2014 : A big German data set associated with isolated and continuous singing.

These datasets have different numbers of gestures, complexity of the gestures, and variety in signers. Not all sign languages, such as British Sign Language (BSL), have standardised datasets, and so researchers often need to source their own data and annotate it.

2.5. Real-Time Sign Language Recognition

Real-time Sign Language Recognition can face challenges that are absent in the case of offline recognition. The requirement for a real-time system is that it must have low latency, with high accuracy, and should be capable to generalize well to changes such as lighting conditions or signer's appearance. Attaining these requirements needs a careful balance between model complexity and computational efficiency.

A major problem with real-time recognition, is getting the system to process video input at a fast enough frame rate that it sees all the individual gestures but doesn't introduce significant lag. It means that there is need to optimize not only the model (architecture) but also the underlying hardware. Techniques like model pruning, quantization and the use of hardware accelerators (GPUs / TPUs) can help improve performance.

There has been a significant progress in real-time Sign Language Recognition models, but they are still challenged to provide the most accurate results where conditions are not optimal such as low light or when signers have obstructed hands. These challenges point to the need for more robust and adaptive models.

2.6. Summary

The literature on sign language recognition seems to reflect a significant advance in the field, especially due to deep learning techniques. Though early systems were dependent on conventional machine learning approaches such as HMMs and SVMs, an increasing influence of neural networks, particularly CNNs, LSTMS, or GRUs, has doubled recognition accuracy and increased robustness. The challenges still remain, mainly with respect to real-time performance and a wide range of gestures representing a particular sign in different languages.

The work in this project is an extension to what has been already done by others on sign language recognition, but of the British Sign Language using GRU-based models. This project seeks to improve sign language recognition systems by securing the limitations provided by existing benchmarks while still building on top of a GRU baseline.

3. Background Research

3.1. Introduction

Sign Language Recognition is a diverse and complicated task where we can identify a lot of research in different types; considering which serve as criteria to evaluate existent method or technology. This chapter discusses the building blocks and developments that support work on automatic sign language recognition systems, with specific reference to British Sign Language (BSL). This chapter will help justify the choice of models, techniques, and tools that are selected for each phase as it progresses into development and implementation.

3.2. Primary Research

3.2.1. Data Collection

30 predefined British Sign Language (BSL) gestures were finalised for use. They were chosen because they are common in daily sign language communication. Data for each gesture was collected by recording videos of myself, process of which has been explained in detail in the methodology chapter ahead.

3.2.2. Data Preprocessing

Key Point Extraction:

After collecting the data, pre-processing was needed to be done for fine-grained feature extraction. Hand, face and pose landmarks were extracted using the MediaPipe holistic model. This step was needed for transforming raw video data to NumPy arrays, a form that can be used to train deep learning models. This process has been explained in detail in the methodology chapter ahead.

The set of 30 gestures were annotated meticulously for each sequence. The data was then split into training and testing sets for model development and evaluation.

3.3. Secondary Research

3.3.1. Sign Language Recognition

Early Approaches:

Template matching and rule-based systems were approaches to sign language recognition for years as it was a research branch. Data in these methods follows a strict protocol that makes them unable to generalize across users and sign variations. Sign language, by its nature is dynamic and continuous; the earlier SLR systems simply gave poor accuracy causing them to be unsuitable for practical application.

Introduction of Machine Learning:

SLR systems were improved significantly by the introduction of machine learning techniques, notably Support Vector Machines (SVMs) and Hidden Markov Models (HMMs). Although these models were more adaptive of variability and complexity of sign language, they still required a lot of feature engineering. As outlined by Starner et al., (1998), the shift from rule-based to statistical approaches led to a major boost in SLR systems accuracy and robustness.

3.3.2. Advances in Deep Learning for Gesture Recognition

Convolutional Neural Networks (CNNs):

Deep learning, especially the introduction of Convolutional Neural Networks (CNNs), revolutionized gestures recognition using automatic feature extraction. CNNs can capture spatial features very well and, in most cases of SLR systems where hand position/shape is involved, CNNS is a popular choice. A study by Molchanov et al., (2016) demonstrated that CNNs can beat traditional machine learning models in static gesture recognition tasks.

Recurrent Neural Networks (RNNs):

We continue to see progress with Recurrent Neural Networks (RNNs) – LSTMs and GRUs – which enabled temporal sequences modelling in gesture recognition. Because LSTMs have shown that they can process sequences, it makes them suitable for capturing long-distance dependencies. However, GRUs are a more efficient alternative to LSTM providing comparable performance, especially in real-time applications.

Comparative Studies:

There are some comparative studies reporting that GRUs are faster in terms of processing and training times, making them better suited for a real time systems even though LSTMs have an upper hand at handling long sequences. This conclusion led to the focus on GRU models in this project when developing the BSL recognition system.

3.3.3. British Sign Language: Unique Challenges and Considerations:

Distinctive Features of BSL:

BSL can be quite difficult; involving two hands and facial expression where body language conveys a meaning. BSL uses spatial relationships as well as non-manual signals like head movements or facial expressions. According to Sutton-Spence and Woll (1999, p. 31), one important aspect of BSL is that it can be complex, and it requires the capturing of these nuances in any recognition system.

Facial and Pose Landmarks:

The use of facial and pose landmarks in SLR systems has been recognized as an important factor for enhancing recognition accuracy. Studies by Koller et al., (2015) say that these additional cues can appreciably improve the system's sign interpretation capability for higher level visual languages like BSL that include non-manual signals . This insight proved to be determinative in the decision to use MediaPipe's holistic model (which returns both hand and face landmarks).

3.4. Summary

In this chapter, a distinction between primary and secondary research activities that led to the development of features within BSL recognition was made. The original research consisted of creating physical data collection, preprocessing. This second part of research increased the theoretical background and justified the options made in selecting distinct models/techniques or approaches to handle problems related to BSL recognition. This dual approach guaranteed that the project was founded on original experimentation as well as established knowledge, leading to a thoroughly researched system.

4. Legal, Social, Ethical and Professional Issues

4.1. Introduction

To create a sign language recognition system, particularly one that seeks to facilitate communication between the people who use and understand natural languages and those using gestural forms of communication (sign language), it is necessary to contemplate legal, social, ethical, and professional issues. The system should technically sound, but also socially responsible, legally compliant and ethically supportable. This chapter discusses the difficulties and considerations in these domains of a BSL recognition system.

4.2. Legal Issues

- Data Privacy:

  In this project, data was collected by recording personal video sequences to capture BSL gestures. Because this data is sensitive in nature, it must abide by privacy laws like the General Data Protection Regulation (GDPR). The data was all self-collected and involved the author, but it certainly followed the principles of GDPR by anonymizing the data and storing it securely. Since the roommate was a part of the final test phase, it was ensured that their participation was strictly voluntary and informed, with clarity regarding how their data would be used and what the purpose of the project was.

- Intellectual Property:

  In this project, the models that were built use self-collected data and independently created code. This reduces intellectual property risks that are associated with third-party datasets or proprietary algorithms. This project relied on the use of external libraries/frameworks such as TensorFlow and Mediapipe which were rightly given credit to ensure compliance with their respective licenses.

- Accessibility Regulations:

  Since the project aims to construct a BSL recognition system, adhering to accessibility standards is one of the major focusses. This system is intended to help make communication more accessible for deaf and hard-of-hearing and

speaking individuals by offering a way to communicate directly with people who do not know sign language. It aligns with the broader legal frameworks that mandate equal access to communication tools and technologies.

4.3. Social Issues

- Inclusivity:

  This project was aimed at breaking the communication barrier between sign language users and non-signers. The system could make various settings like education, public services and social interactions more inclusive by offering a tool that can translate BSL into text in real-time. This project demonstrates societal responsibility as it fulfils to a particular need of the Deaf community and contributes to better integration in society.

- Cultural Sensitivity:

  British Sign Language isn't just a language, it's also an integral part of Deaf culture. All decisions during this project were made with the greatest respect for BSL, its nuances and cultural importance, keeping in consideration the importance of such a tool for facilitating communication for ALS and Parkinson's patients. The recognition model was trained with gestures which are practical in everyday communication and are used more frequently.

- Impact on Human Interpreters:

  One social concern is the influence of SLR system on the BSL community. Although the technology is supposed to lessen the communication gaps, there needs be a way where it does not accidentally further disenfranchise sign language users by subbing in human interpreters or reducing reasons for non-signers to learn sign language. This new system should be considered an assistive technology and not a replacement of human interaction.

4.4. Ethical Issues

- Informed Consent:

  The participation of the roommate and friend for real-time testing called for informing them of the purpose of the project and how their data was indented

to be used. This ethical approach allowed for voluntary informed consent, maintaining respect and autonomy.

- Bias and fairness:

  There is a possibility of bias because the data is self-collected (especially if the gestures made are not representative of how others might perform them). Although this method ensures consistent and controlled data collection, it is considered as a limitation with suggestion that future work should be done on the model with other datasets that are more diverse Fairness in the predictions made by the model was very important, especially during evaluation when looking at accuracy scores across different gestures.

- Transparency and Accountability:

  The project ensures transparency about the capabilities and limitations of the BSL recognition system. The accuracy and possible errors of the system have been documented making sure that users would have realistic expectations. Given the delicate nature of this use-case, transparency is crucial for both trust and being responsible around how it gets used.

4.5. Professional Issues

- Adherence to Standards:

  The project adhered to standard software development practices — i.e. code documentation, version control and thorough testing. Over time, they also led to a much more robust and maintainable system. Moreover, it was based on reliable technologies using well-established libraries and frameworks.

- Continuous Learning:

  With rapid advancements in machine learning and computer vision, staying up to date with the latest tools and techniques was necessary. It involved constant study, and especially when it came to learning about GRU models — an architecture not as popular in sign language recognition tasks that use LSTMs or CNNs. This exploration of GRUs exhibited a desire of commitment and making improvements in what is already happening in the field.

- Engagement with the Community:

  Although this was a project that began with independent, it is important to note its reliance upon the support of several communities. Future work will focus on co-design and involving the Deaf community from a wider perspective to ensure fulfilling real-world expectations. This involvement is vital for the continued growth and success of this system.

4.6. Summary

This chapter addressed the legal and social, ethical as well as professional issues faced during the design of a BSL recognition system. Taking appropriate permissions under the data privacy laws, addressing intellectual property and ownership issues, adhering to accessibility regulation was amongst some key legal compliances which this chapter tactfully cleared. It aimed to improve inclusivity and to respect special cultural nuances associated with BSL, keeping in mind social responsibility. From an ethical perspective, the project was grounded on informed consent, fairness and transparency while also taking any biases in data into account. From a professional standpoint, the project was executed within some best practices and showed continued learning with an importance of community engagement. These were crucial issues to answer, in order to develop an accountable system not just programmatically but socially and ethically as well.

5. Methodology

5.1. Introduction

The methodology chapter summarises how a structural process was implemented and developed for designing a sign language recognition system. In this project where I worked with BSL (British Sign Language) gesture recognition, a well-defined workflow was essential which consists of data collection, preprocessing, model building and evaluation. The idea was to have a speedy and robust real-time gesture recognition system. This chapter starts with a description of the data collection methodologies and then explains how the data were preprocessed to prepare for model building. In subsequent sections we detail the model development and training processes, centred mainly around GRU (Gated Recurrent Unit) which was our most effective model. The methodology also involves discussion of the hyperparameter tuning and optimisation techniques which have been able to improve the performance of the model. This chapter lays the groundwork for understanding how development and implementation are discussed in following chapters.

5.2. Project Set-up and Environment Configuration (Installing Dependencies & Project Structure)

The project started by creating an environment that was necessary to develop and test the sign language recognition system. That included installing and managing dependencies. The main libraries that were used in this project include TensorFlow, OpenCV & Mediapipe for implementing computer vision pre-processing techniques, NumPy being a primary asset to seamlessly apply mathematical operation, and Scikit Learn for evaluation. To make the experience even smoother, a centralized dependency installer file imports.py was created to manage these dependencies through the rest of your project modules. This file worked as a central source for importing libraries, avoiding redundancy and inconsistencies due to version mismatch or missing dependencies.

The project was structured to differentiate different parts of the projects, namely data collection, preprocessing and model building and then real-time application. This modularised approach allowed for simpler debugging, testing and future changes.

5.3. Data Collection of BSL Gestures

As data collection was essential in the project, without enough and appropriate level of quality, model performance would be worse.

5.3.1. Folder Set-Up

First, a directory called "BSL_Data" was set up using the os python library where all data would be stored. A NumPy array was initialised that matched the list of 30 sign language gestures/actions selected for use in this project. 30 video sequences were to be recorded per gesture where each sequence would consist of 30 frames ensuring that sufficient data was available for training and the input size remained uniform throughout the dataset. A for loop was used to create a directory with a nested folder structure for each sequence: BSL_Data/gesture_name/sequence_number. Each sequence file would contain 30 frames in the form of .npy files, which are binary files used to store NumPy arrays.

Figure 1: Structure of Data Files

5.3.2. Function Creation

Once all the folders were set up, 3 functions were created which were as follows:

- mediapipe_detection
    - When OpenCV reads frames, they are in BGR colour format. This function converts the frame from BGR to RGB format using the cvtColor method of OpenCV.
    - This is because when the frame is fed into the mediapipe model that detects landmarks, it expects the RGB format.

- Once the landmarks are detected, the frame is converted back BGR format as OpenCV will be used to display the image later on.
- The detected landmarks are stored for further use.

- custom_landmarks
  - This function checks whether the landmarks were detected for each part (face, right-hand, left-hand and pose), then draws points and connections between them.
  - The drawings have been customised using drawing utilities provided by Mediapipe to have more control over the appearance such as colour, thickness and circle radius.
  - An example of the total landmarks detected on the face, hands and pose can be seen below:

- extract_keypoints
  - This function extracts x ,y, z coordinates and visibility(for pose) of all the detected landmarks.
  - A list of lists is created using NumPy where each inner list contains the attributes of each landmark.
  - The .flatten method of NumPy Arrays converts this list into a flat one-dimensional array.
  - If landmarks on any part are not detected, this function fills it with zero-fill array of the same size as the total expected keypoints for that part:
    - Pose: 33*4=132 keypoints (since pose has an extra visibility attribute)
    - Face: 468*3=1404 keypoints
    - Right Hand: 21*3=63 keypoints
    - Left-Hand: 21*3=63 keypoints
  - All these keypoints are then concatenated into one single flat NumPy Array.
  - This function is generally considered part of preprocessing, but it has been defined and used for data collection for the purpose of immediate verification of detection.

These functions were to be used in multiple scripts; hence they were defined in a separate file simplifying workflow and reducing redundancy.

5.3.3. Data Collection

30 BSL gestures which were initially selected for use were recorded by filming videos of myself performing all those action 30 times.

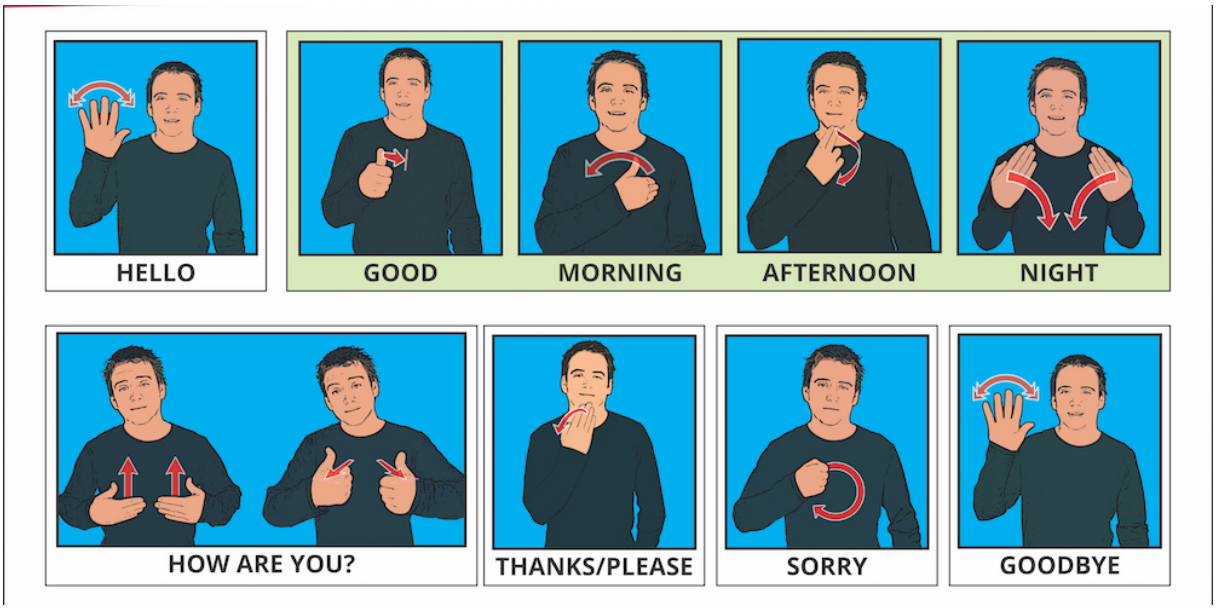

Figure 2: British Sign Language Gestures (British Greeting Signs)

First, a path was set to save all the collected data in "BSL_Data" folder which was created earlier. A NumPy array of string was defined where every string represented the gesture that was to be recorded. The number of sequences and number of frames per sequences were set(30 for both). This meant that each gesture was repeated 30 times. And each sequence would consist of 30 frames.

Recording of the gestures was done with help of the video capture object using OpenCV. Mediapipe's holistic model was initialised with minimum detection and tracking confidence set to 0.5. This model is capable of detecting landmarks for the face, pose and both hands simultaneously. The confidence sets a minimum threshold for detecting landmarks and a minimum confidence for tracking the landmarks between frames.

With the help of a for loop, 30 sequences of each gesture were recorded. Before commencing each gesture's recording, a 2 second wait time was applied. The screen displayed which action, and which sequence of that particular action was being recorded.

Figure 3: Recorded Gestures for Data Collection

Figure 4: Recording Gestures for Data Collection(2)

It was ensured that frames were accurately captured for each sequence, with the help of checks with if statement that returned error warnings, skipped the rest of the loop and moved to the next iteration.

While recording the gestures, the holistic model detected landmarks with the mediapipe_detection function, drew the customised landmarks with the custom_landmark function and, the extract_keypoints function extracted coordinates, converted them into flat one-dimensional arrays and saved them into the allotted locations in the directory created at the start.

5.3.4. Challenges and Considerations

A major difficulty encountered during the data collection process was replicating gestures consistently. Differences like the position of the hand, light settings or execution speed would create inconsistencies in the dataset. In order to address these problems, each gesture was repeated 30 times, alternating hands and moving the head position each time.

5.4. Preprocessing

Once the data was collected, it needed to be preprocessed so that there was an appropriate dataset on which models could be trained. One crucial step, extracting keypoints was performed while collecting data. This had created and stored an array of the coordinates of all landmarks. The other preprocessing steps performed were:

- Label Mapping

  A dictionary was created to map each gesture to a numerical label. For example: ‘hello’:0, ‘thanks’:1, ‘how_are_you’:2….etc.

- Feature Sets and Labels

  2 lists were created, feature_sequences to store the keypoints for each gesture and target_labels to store their corresponding numerical label. This was done using a for loop to iterate over all 30 gestures and the label_mapping disctionary created earlier.

5.5. Model Development

5.5.1. Exploration of Various Models

In the first stage of model development, it was attempted to explore the difference in accuracy in the models mentioned below. Since LSTM based models are extensively used, and GRUs tend to provide an efficient alternative

with similar performance, I attempted to build a good enough model with performance that can compete with the LSTM based models. Surprisingly, after running a simple accuracy check with the same hyperparameters, GRU seemed to display the highest accuracy. A brief description of the same is as follows:

- LSTM(Long Short-Term Memory)

  An LSTM model was tested, as it's quite popular in sequence modelling tasks. It was, however, performance limited with an accuracy of 17.78%.

- Bidirectional LSTM

  A Bidirectional LTSM was tested to help the model understand context. It processes input data in forward and backward directions, developing precise frame of sequences. It provided an accuracy of 51.11%.

- GRU (Gated Recurrent Unit)

  GRU was tested as it provides a more effective alternative with similar performance, handles sequence data well and is computationally less intensive than LSTM. As the tests showed promising results, the GRU model was chosen as a base for this project.

- CNN-LSTM Hybrid

  A hybrid model that leverages Convolutional Neural Networks (CNN) with LSTM was experimented as well. CNN layers extract spatial features, while the LSTM layers handled temporal characteristics of data. It performed well among others with ana accuracy of 51.17%.

Observing a slighter higher accuracy in the GRU model led to research regarding the possible reasons as to why it could perform better than an LSTM based model. According to studies conducted by Cahuantzi, Chen and Guttel (2021), GRUs generally outperform LSTMs when dealing with low complexity sequences as they have a simpler architecture and faster training times. This led to curiosity about whether tuning its hyperparameters could enhance its performance further. Based on this I decided to develop my model using GRU.

5.5.2. GRU Model Optimization

Having obtained promising chances of performance, optimizing the model was next step. This involved:

- Hyperparameter Tuning

  GridSearchCV was used for an exhaustive search over specified parameter values to discover the optimal hyperparameters such as number of GRU units, dropout rate, dense layers, learning rates and batch size.

- Training

  The preprocessed data was trained using categorical cross entropy as the loss function and Adam optimizer. The hyperparameters concluded in the previous stage were used. The training procedure also included tracking of validation loss and early stopping to prevent overfitting.

- Performance Metrics

  Performance of the model was calculated as accuracy, loss, precision, recall, f1 score and confusion matrix. Being compared against all these metrics allowed for a complete understanding of how well the model did across every gesture.

5.6. Implementation

5.6.1. Integration with OpenCV for Real-Time Gesture Recognition

The trained GRU model was combined with OpenCV for real-time gesture detection. OpenCV was used for capturing live video from the computer's camera and processing it frame by frame. The combinations of these keypoints of each frame were sent as input to the GRU model and it predicted the gesture corresponding to them.

This pipeline could manage the input from the camera, process frames and extract keypoints from them, use these key points as reference for the trained model and display the identified gesture on screen. The real-time recognition system was optimized for running on a normal laptop to with minimal latency and smooth performance.

5.6.2. Custom Visualisation with MediaPipe

In order to make the user experience more interactive, Mediapipe library provided some powerful drawing tools which were customized in this project to visualize all visible landmarks in the live video feed. Coloured and line thickness were utilized to help the user distinguish between face, hand and pose landmarks.

This visualization gave us an insight not only on how the model makes predictions, but it also helped as a debugging tool over the development process. This visual feedback allowed to verify that the keypoints were detected correctly and were processed properly before going forward to the model.

5.7. Evaluation & Testing

5.7.1. Evaluation Metrics

The complete set of accuracies, confusion matrix, precision, recall and F1-score were studied for evaluating the final GRU model. These values were computed on the test data which was 20% of original dataset reserved for evaluation.

The confusion matrix showed that some of the gestures that were recognised, and others were struggling. These metrics of the precision, recall and F1-score were quite vital for analysing model performance for imbalanced class where gestures had varied examples.

5.7.2. Testing the Performance under Various Conditions

The implementation was tested for robustness and reliability under various common real-world scenarios. The system was tested in a variety of lighting conditions, with different backgrounds and users to generalise well past the training data.

These tests showed the system's ability to identify gestures quickly, with very minimal false positives. Generally, there was consistency across gestures in

different conditions, but a few gestures faced some misclassification. These findings were documented to possibly be developed further in the future.

5.8. Summary

In summary, the methodology chapter described how a BSL recognition system was designed through an elaborate process. From the collection of gesture data, preprocessing and feature extraction to implementation and evaluation, everything was covered. In the development phase, multiple machine learning models were tested for this task, and it was discovered that the GRU model is most appropriate one. The model was fine-tuned and optimized with respect to required hyperparameters, ensuring the system delivered incredible accuracy & performance in run-time. This systematic approach supported the hope for success of this project and a well-designed path for further advanced requirements too.

6. Development of the Model

6.1. Introduction

Creating an accurate and efficient sign language recognition model is a challenging multistep process that deals with different constraints, from choosing suitable base architecture to preprocessing data and tuning of hyperparameters. This chapter has described the individual steps followed in creating, training and fine-tuning of these systems, with deep learning as the basic focus.

This chapter has further described the training and validation step, performance of several models in terms of loss functions and finally results obtained using best-optimized GRU model. It includes what steps have iteratively helped the model during its development to perform and a visual understanding of the model's performance. Finally, what challenges were faced and how they were tackled in order to make the model more accurate and reliable will form an end note for this chapter.

6.2. System Optimization and Final Model

6.2.1. Hyperparameter Tuning

The GRU model was chosen as the base architecture and benchmark after achieving better performance with it in initial trials. A set of hyperparameter tuning experiments followed aiming for higher accuracy, precision, recall besides generalization power. The aim was to find the perfect setup in order for the model to learn well but not overfit or underfit. The experiments conducted are summarised below:

- GRU Layers and Units:

  The number of GRU layers, as well as the units per layer were tuned to see how the model learnt complex patterns. The learning capacity of the model can increase with more units and layers, but this can also lead to overfitting.

- Dropout Rate:

  To avoid the problem of overfitting, dropout regularisation was applied between layers. Setting dropout rate too high can help in generalisation but can

also result in poor performance, because once it reaches a very large value of dropout can hinder learning.

- Dense Layers:

  After the GRU layers, dense layers were included to map learned representations to output space. This was employed by adjusting the number of units as well as the activations function for optimisation.

- Learning Rate:

  The learning rate tells the model how quickly it should adapt to any change. A low learning rate ensures that the model will converge to a stable solution — however, it may take longer— while a higher one could be faster but overshoot the optimal solution.

- Batch Size:

  It is a hyperparameter that affects the stability of gradient descent. Smaller sizes can be quite noisy but often generalize better.

- Epochs and Early Stopping:

  We also changed the epochs range with early stopping to make a halt on training when we did not gain much improvement.

Experiment	Observations
▪ GRU Layers/Units: 128 Units (1 Layer) ▪ Dropout Rate: 0.5 ▪ Dense Layers: 64 Units (1 Layer) ▪ Learning Rate:0.0005 ▪ Batch Size: 32 ▪ Epochs: 100 ▪ Accuracy: 20%	Overfitting; low confidence

▪ GRU Layers/Units: 128 + 64 Units (2 Layers) ▪ Dropout Rate: 0.5 ▪ Dense Layers: 64 Units (1 Layer) ▪ Learning Rate: 0.0003 ▪ Batch Size: 16 ▪ Epochs: 150 ▪ Accuracy: 36.11	Improved generalization, some classes still struggle
▪ GRU Layers/Units: 256 + 128 + 64 Units (3 Layers) ▪ Dropout Rate: 0.4, 0.5, 0.5 ▪ Dense Layers: 256, 128 Units (2 Layers) ▪ Learning Rate: 0.0001 ▪ Batch Size: 16 ▪ Epochs: 200 ▪ Accuracy: 27%	Increased model complexity, struggles with overfitting
▪ GRU Layers/Units: 128 + 64 Units (2 Layers) ▪ Dropout Rate: 0.4, 0.7 ▪ Dense Layers: 128 Units (1 Layer) ▪ Learning Rate: 0.0001 ▪ Batch Size: 64 ▪ Epochs: 200 ▪ Accuracy: 4%	Underfitting, predictions spread across different classes
▪ GRU Layers/Units: 128 Units (1 Layer) ▪ Dropout Rate: 0.3, 0.4 ▪ Dense Layers: 64 Units (1 Layer) ▪ Learning Rate: 0.0003 ▪ Batch Size: 32 ▪ Epochs: 300 ▪ Accuracy: 78%	Good generalization, reduced overfitting, concentrated predictions
▪ GRU Layers/Units: 128 Units (1 Layer) ▪ Dropout Rate: 0.3, 0.4 ▪ Dense Layers: 64 Units (1 Layer) ▪ Learning Rate: 0.0004	Good generalization, slight degradation due to faster convergence

▪ Batch Size: 32 ▪ Epochs: 300 ▪ Accuracy: 75%	

Table 1: Results of Various Hyperparameter Values

Every experiment offered a key learning regarding the impact of different hyper-parameters over model learnability & performance:

- GRU layers/units:
  The inclusion of GRU layers and units as the model gains information increased learning capacity, however, ultimately it overfitted due to a complicated architecture.
- Dropout Rate:
  This helped to control overfitting. But discovering a proper rate of dropout was difficult; too low caused overfitting, and too high hindered the learning process.
- Dense Layers:
  Addition of dense layers here with regularization helped to map the learned features effectively to the output space. But when there were many dense units or layers it led to overfitting.
- Learning Rate:
  The learning rate was chosen under the consideration that a smaller one could potentially help to stabilize training and result in better accuracy.
- Batch Size:
  Smaller batch sizes seemed to do better in general, but more iterations per epoch were needed.
- Epochs and Early Stopping:
  By keeping track of the validation loss with early stopping, this model could stop training when it reached where there was no more improvement, preventing overfitting.

6.2.2. Optimization with GridSearchCV

After experimenting with tweaking hyperparameters, I stumbled upon the concept of GriSearchCV and decided to try it for the model. With the first grid search giving an accuracy of 84.44%, it was decided to increase the number and range of hyperparameters used in the gird search, hoping that this would help find an even more optimal set of hyperparameters that would nudge the model accuracy closer to 95%.

6.2.2.1. Initial Grid Search and Best Parameters

While. Grid-Searching with narrower hyperparameters, these were the best configuration:

- Learning Rate: 0.0001
- Dropout Rate: 0.3
- Batch Size: 32
- Epochs: 300
- GRU Units: 128
- L2 Regularization: 0.005

The model tuned with this configuration was able to differentiate between the different gesture classes accurately at a mean accuracy of 84.44%, with minimal overfitting.

6.2.2.2. Expanded Grid Search

The additional computational burden of running such an expanded grid search was significant. This line of thinking was fine when the task had only a few hyperparameters with clear ranges, but as models grew larger and more complex, the search space increased exponentially. There were 7776 combinations of hyperparameters that had to be assessed by the expanded grid. Despite using multi-core processing (n_jobs=-1), the grid search ran for three days and was still incomplete.

That extended runtime was due to the large parameter space. It became impractical to continue considering the time constraints of this project.

6.2.2.3. Challenges with Expanded Grid Search

The additional computational burden of running such an expanded grid search was significant. This line of thinking was fine when the task had only a few hyperparameters with clear ranges, but as models grew larger and more complex, the search space increased exponentially. There were 7776 combinations of hyperparameters that had to be assessed by the expanded grid. Despite using multi-core processing (n_jobs=-1), the grid search ran for three days and was still incomplete.

That extended runtime was due to the large parameter space. It became impractical to continue considering the time constraints of this project.

6.2.2.4. Strategic Decision to Revert to Original Configuration

Given the time and resources required for a larger grid search, it was decided to halt the search. Instead, choosing the results of the first grid search seemed more practical as they provided a strong enough balance between performance and feasibility. This method allowed to proceed onward with confidence that the model would generalize well on other datasets.

6.3. Summary

The sign language detection model was developed with careful design, iteration and optimisation . At first, a GRU-based model was introduced to capture the temporal dependencies of gesture sequences. The model architecture was refined using layers such as Bidirectional GRU, LayerNormalization and Dropout for better performance. A grid search was performed to further optimize the model parameters which enhanced the performances greatly.

7. Implementation and Testing

After performing a grid search, which gave a solid foundation, a few random changes in parameters were explored . Surprisingly, those unplanned changes made a considerable increase in model accuracy. The new parameters are as follows.:

- GRU Units: 256
- Dropout Rate: 0.25
- Learning Rate: 0.00001
- L2 Regularization: 0.001
- Dense Units: 128
- Batch Size: 16
- Epochs: 300

These parameters were found not only through a systematic grid search but also continuous trial and error until an optimal combination of hyperparameters showed improved performance.

7.1. Final Model Architecture

The final model consisted of the following architecture:

- Input Layer:

  The input layer was setup to receive sequences of shape (30, 1662), where 30 meant sequence length (number of frames), and the 1662 was the number of features extracted per frame.
- Bidirectional GRU Layer:

  A bidirectional GRU layer with 256 units was used. The GRU units are bidirectional; they process the input sequence in both forward and backward directions increasing the capability to capture temporal dependencies from the gesture sequences.
- Normalization:

  To stabilize training and improve convergence, A LayerNormalization layer was added just after the GRU Layer.
- Dropout Layers:

  To avoid over-fitting, two Dropout layers were used:

  A Dropout layer with rate 0.25 was added after the GRU layer.

The next Dropout layer that was added after the dense layer had a rate of 0.25 to reduce chances of overfitting.

- Dense Layer:

  To map the learned features to the output classes, aa dense layer of 128 units with ReLU activation, and L2 regularization set to be at (0.001) was added.

- Output Layer:

  The last dense layer applied a softmax activation function to generate a probability distribution on the 30 gesture classes.

The model was compiled with the Adam optimizer, known for efficiently handling large-scale data. The grid search and subsequent testing suggested the use of a very low learning rate of 0.00001, which enabled the model to converge slowly and steadily over 300 epochs.

The model was fit on the training data with a 20 % validation split to continuously evaluate its performance on unseen data.

7.2. Real-Time BSL Detection

Once the GRU model was built and refined, the next important step was to deploy real-time detection with which the performance of the optimized/pre-trained model could be tested in a practical scenario. This section discusses how the GRU model was integrated into a real-time detection model with help of cameras to collect live data and the Mediapipe library for detecting key points.

Loading the Trained Model:

The first implementation task was to load the pre-trained GRU model (final_model_gru_best_params.keras). This model, which was trained and fine-tuned on the dataset was now ready for real time prediction.

Setting Up Video Capturing:

The video feed was captured with OpenCV's "cv2.VideoCapture" function. this function accessed the camera on the system. A short break was introduced t so the camera and model could both be initiated.

Mediapipe for Keypoint/Landmark Detection:

For extracting keypoints from every frame, the Mediapipe holistic model was used. This model accurately detects and tracks multiple landmarks:1) face; 2) hands; and, 3) body pose. These landmarks are very important because they are the input features for the GRU model.

Processing Each Frame:

- Frame Conversion:
  Every frame from the video feed was converted from BGR to RGB colour space, which is a requirement of Mediapipe.
- Landmark Detection:
  The frames were processed by the holistic model to detect all landmarks. Those landmarks were then drawn onto the frame for visual reference.
- Keypoint Extraction :
  From the detected landmarks, , 4 types of keypoints were extracted; face, pose, right hand, left hand. For the landmarks that were not detected, zeros were padded so as to maintain a consistent size of input for the GRU model.

Sequence Buffer:

A sequence buffer was maintained to hold 30 frames which ultimately forms the data passed on to GRU model as it expected a sequence of 30 frames. Over time this buffer kept getting refreshed when newer frames were captured. A confidence threshold of 60 percent was also included.

Real-Time Prediction:

Once the sequence buffer contained 30 frames in it, it was sent to the GRU model for prediction. The model predicted the most likely action class and confidence score. When the confidence score was at least 0.6, we displayed the predicted action on a live video feed.

Displaying the Output :

The predicted action along with the confidence score was displayed in text on video feed using OpenCV's "putText" method. It gave instant feedback on the model's predictions.

Graceful termination:

The real-time detection loop continued until the user pressed 'x', where the video capture was released and closed all OpenCV windows.

7.3. Summary

This chapter demonstrated progressing from model development to real-world application: the trained GRU model being implemented live for sign language detection. This included loading the pre-trained model as required, setting up a video capture system and utilizing the Mediapipe Holistic Model for detecting keypoints in the live feed. A sequence buffer was included to store the latest frames before they could be used for real time prediction. The implementation displayed the ability of the model to predict actions and transform them into displayed text on the video feed. This integration provided a positive conclusion about the performance of the model in a practical setting, which would be followed by evaluation and analysis in the next chapter.

8. Evaluation and Analysis

8.1. Introduction

This chapter presents a detailed discussion about the performance of the final GRU model and the Real Time British Sign Language Detection Model. The evaluation includes how well the model predicts, detects and acts in terms of accuracy, precision, recall along with its general performance under controlled testing environments as well real time dynamic environments. This assessment is vital to understand the strengths and weaknesses of the system and analysing the areas for improvement in the future.

8.2. Results and Visualisations of GRU Model

After training the model, it was tested on the test data and succeeded in reaching an accuracy of 90.00%.

Detailed results, including visualizations are as follows:

Accuracy and Loss Curves

Training and validation curves below indicated a steady improvement in accuracy and a consistent reduction in loss over epochs, showing that the model converges well with less likeness for overfitting.

- Accuracy: Model accuracy is an evaluation metric that shows the percentage of correct predictions made by the model out of all the predictions. Training and validation accuracy both increase along a steadily upward curve and eventually converge, reaching a final accuracy of 90.00% on the test set. This also indicates that the model does not overfit
- Loss: Model loss is an evaluation metric that tells us how far off the model's predictions are from the actual labels. Both the training and validation loss curve slopes down, indicating that the model was indeed effectively learning. The validation curve does not increase, which shows that the model is not overfitting to the trained data, suggesting that it is generalising well.

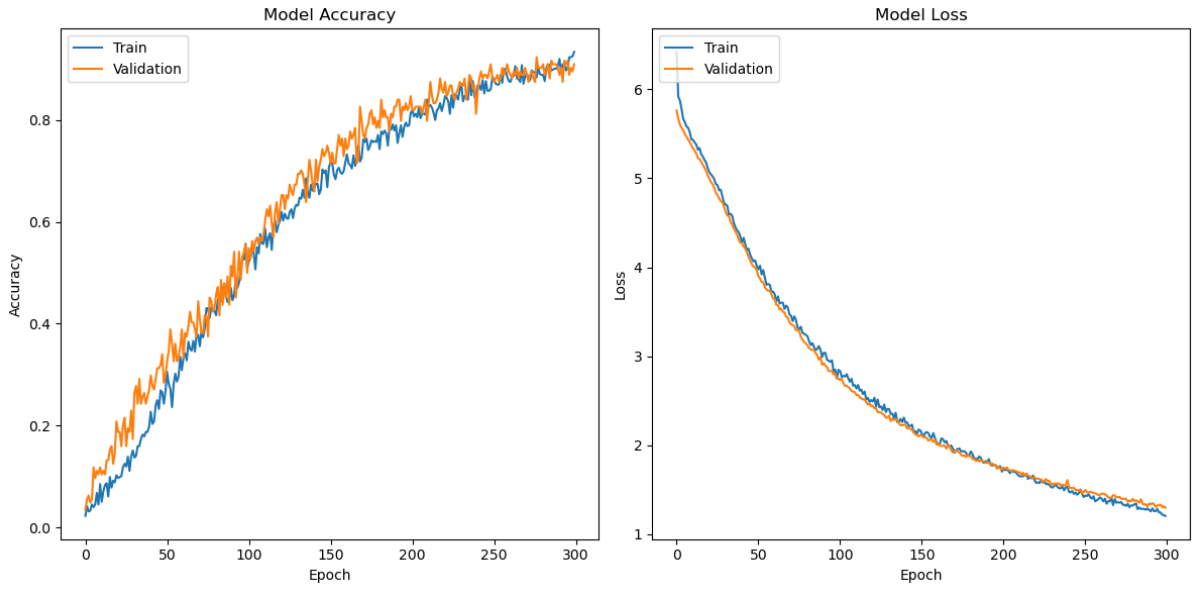

Figure 5: Accuracy and Loss Curve

Classification Report:

The model showed high precision and recall in most of classes including some class having perfect scores. The average weighted precision was 88% and the recall was 89%. The model appears to detect true positives and minimise false positives, indicated by the balanced F1-scores.

For example, the model achieved the following metrics for key classes:

- **Thanks**: Precision 1.00, Recall 0.86, F1-Score 0.92
- **How Are You**: Precision 1.00, Recall 0.92, F1-Score 0.96
- **Danger**: Precision 1.00, Recall 1.00, F1-Score 1.00

Confusion Matrix:

A confusion matrix is the comparison of the true labels with the predicted labels after the model has been trained. Higher values on the diagonal of the confusion matrix indicates that more of the predictions fell into the correct class with least misclassification. This confirmed that the model was quite robust in detecting and recognizing the gestures properly.

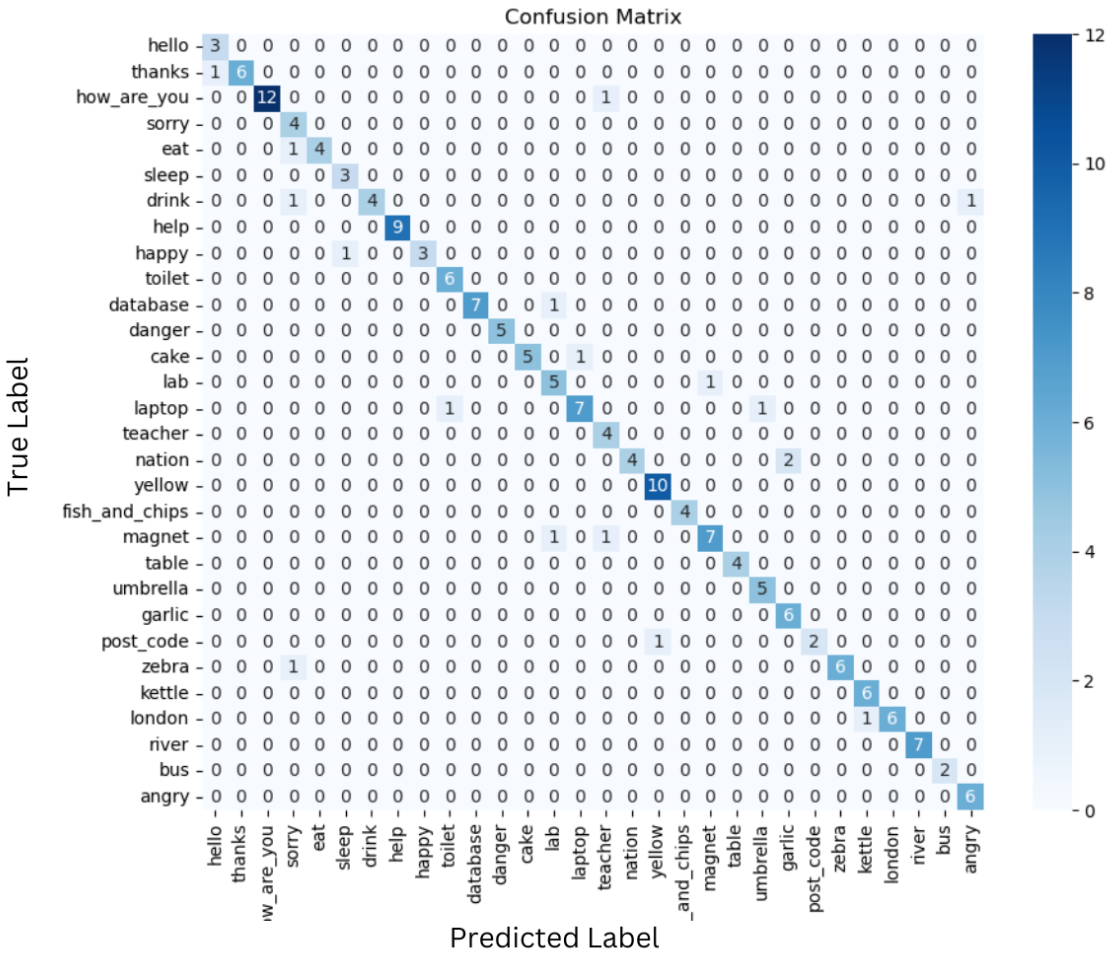

Figure 6: Confusion Matrix

8.3. Critical Review

Finally, real-time application of the GRU model put its robustness to a test. Running the model in a live setting exposed several practicalities and learning as well.

Latency:

One of the major challenges faced was latency in predictions made by the model during real-time testing. In a real-world use-case scenario, the time between performing an action and viewing its predicted result on screen could be too disruptive. There are many possible reasons why this latency:

- The computational complexity of processing each frame.
- The heavy load of running the model on available hardware in real-time.
- Potential inefficiencies in the data pipeline or video capture process.

Camera Angle Sensitivity:

The accuracy of the model was highly affected by camera-specific angles. Some gestures required some slight change of camera angle or subject placement adjustments to make the predictions correct. One reason for this is that the training data may have lacked diversity in terms of camera angles, which makes it a sensitive model, working well only under certain conditions.

Misclassifications and Consistency:

Many misclassifications were observed while testing the model in real-time with some gestures particularly:

- "Bus" was occasionally confused with "Hello" and vice-versa.
- "Eat" was often misclassified as "Thanks"
- "Garlic" was always confused with "Thanks"
- Some that were commonly misclassified or not detected at all: “Database”, “Danger “and “Umbrella”.

This misclassification could be caused due to the following reasons:

- Lack of training data for some gestures.
- Overlapping features of similar gestures causing the model to be confused.
- Difference in gesture execution between training and real-time scenarios.

Different Predictions for Different Users:

The model seemed to give varied predictions for the same gestures when performed by different users. For example, garlic was always misclassified as hello for user 1 but was accurately predicted when user 2 performed the gesture. Screenshots for the same are below:

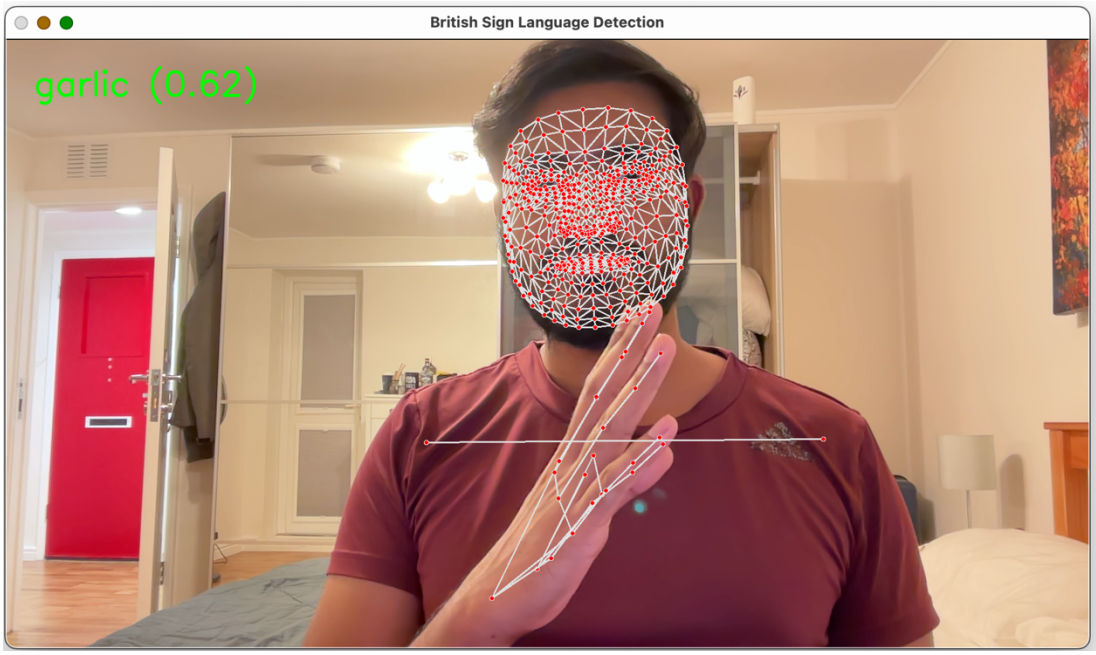

Figure 7: Real-Time Sign Language Recognition with Different User (1)

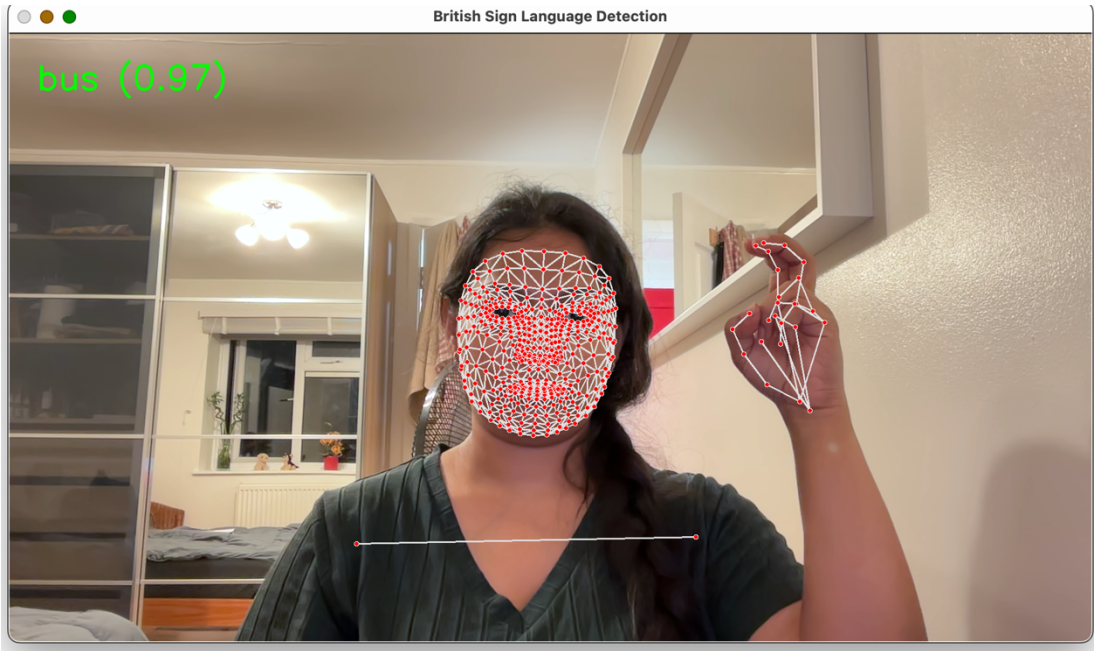

Figure 8: Real-Time Sign Language Recognition with Different User (2)

Real-Time Usability:

However, in real-time, the detection model showed promising performance by correctly identified different gestures with high confidence. Yet, overall, the system is limited due to its high latency and demand for a specific environment for every prediction.

Several important takeaways have been observed from comparing the model's controlled offline setting performance with its real-time one:

Accuracy:

The offline model does well in classifying the data with an accuracy of 90% but this high classification accuracy was not completely successful when it came live. Not only this but the real-time system was also prone to classification errors and needed very specific conditions to give accurate results.

Generalization:

It was tested whether the model could generalize from training data to real-world application, which led to identifying that even though it generalizes well in a static environment, natural variations introduced during run-time were not handled consistently.

Usefulness:

Implementing the model in real-time indicates that it can potentially be used for practical use but manages to highlight the gap between the model performance in an offline setting and a real-world application. A problematic latency and camera angle sensitivity were discovered, both of which pose challenges to the usability but have the potential to be solved in the future.

8.4. Challenges and Observations

Multiple challenges were encountered throughout the project that led to numerous iterations for improvement and refinement. Key challenges and solutions are summarized below:

- FACEMESH_CONTOURS vs. FACE_CONNECTIONS:
  Originally the model tried to use FACE_CONNECTIONS for rendering facial landmarks. But this method did not work accurately. To overcome this, FACEMESH_CONTOURS was used which was more successful and efficient at detecting the facial landmarks.

- Camera Index Identification:

  One of the more difficult issues faced getting the right index for camera. This caused a big problem as each system may have multiple cameras with different indexes. To solve this issue, a script was created that would loop over potential indices to determine which camera was active at the that moment. This solution simplified the process and ensured that the correct camera was selected for use.

- False Warnings for Imports:

  Many imports gave out warnings, especially the ones with third party libraries like Tensorflow Keras. Warning of imports not in use had appeared even thought that was not the case. After chasing all possible solutions to solve the issue, it was concluded that those warnings did not seem to impact the code. Everything else also worked as designed, so the project proceeded without any problems.

8.5. Comparison and Discussion

A paper by Dhulipala, S., et, al. (2022), compares the performance of an LSTM model and a CNN model which trains on 30 sign language gestures and predicts them when performed in real time. This paper is appropriate for comparison of the GRU model in this project due to the similar objectives, motivation, dataset size and evaluation metrics.

Their results concluded that their CNN model outperformed their LSTM model in predicting British Sign Language. It showed better accuracy, precision and reliability compared to their LSMT model. The GRU model developed in this project managed to outperform their LSTM model significantly but could not provide better results than their CNN model. The results of all 3 models are as follows:

MODEL	ACCURACY	PRECISION	RECALL
CNN	98%	97%	96%
LSTM	49%	52%	37%
GRU	90%	88%	89%

Table 2: Result Comparison of Various Models

The GRU model outperformed the LSTM model due to having fewer parameters because of the combined gates. The LSTM model had 3 gates, the input, forget and output gates which required more parameters to be tuned.

CNN excelled at extracting spatial features using convolutional filters that can capture patterns likes edges, shapes and textures. Meanwhile the GRU model is meant to handle temporal dependencies. This does prove helpful in recognising the order of the gestures but does not capture the complex spatial features of each frame as effectively as the CNN model does.

8.6. Summary

Results of the final GRU model in an offline setting as well as real-time testing have been shown in this chapter. The model performed well in a controlled environment (high accuracy and great classification metrics). However, the real-time implementation presented many major problems like latency, camera angle sensitivity and occasional misclassifications. The results compared with experimented LSTM and CNN models with similar datasets concluded that the GRU model outperformed the LSTM model but did not succeed in providing better performance compared to the CNN Model. While results indicate that while the model offers great potential, they reveal larger issues requiring improvement to increase its level of reliability and usability in practice. Findings from this evaluation will help to improvements in the future.

9. Conclusion and Future Work

9.1. Conclusion

This project aimed to develop a system for BSL (British Sign Language) Recognition based on deep learning models. The initial phase was marked by an investigation of different models and technology, which eventually led to constructing a GRU-based model to achieve accurate sign language recognition. An exhaustive grid search was performed to find the optimum combination of hyperparameters.

The final GRU model had an accuracy of 90%, after several iterations and refinements. This model achieved high precision and recall in recognising a variety of BSL actions demonstrating its generalization across different classes. When the model was implemented in real-time it proved to be successful in many areas, but it brought a few challenges regarding latency and camera angle sensitivity. While these do not brush aside the accuracy of the model, they do highlight practical complications and how challenging the real-world deployment can be. Furthermore, the model had difficulty in real-time testing distinguishing some gestures (such as Garlic and Thanks) with overlapping characteristics. It will be extremely important to address these misclassifications in order for the system to become more reliable.

In this regard, the project appears to have done what it set out to do successfully, namely showing that is possible for a BSL sign language recognition model using deep learning (particularly GRU networks) can perform well. This not only attests to the performance of the model in controlled settings, but also indicates that there is possibility for an improved real-time application with a few adjustments.

9.2. Future Work

Although the current model seems to be quite successful, some areas for improvement and are:

Model Optimization:

Although the GRU model is able to achieve high performance, further tuning can significantly improve both accuracy and efficiency. Further research in this field may also study hybrid models that utilize convolutional layers of CNN to extract spatial

features and temporal sequence modelling of GRUs. It could help in improving real-time gesture recognition and reducing misclassification rates.

Latency Reduction:

The real-time implementation faced some latency which can be improved in further versions. Latency reduction is a top priority for future development. Effort to optimize the computation pipeline and utilize better hardware or approach for frame processing can lead in reducing delays resulting into practical model as a real time system.

Expanding the Dataset:

The current model was trained on a limited set of gestures. In case of real-time misclassifications and camera angle dependency, larger datasets comprising of a diverse set of gestures may lead the future research path. This might mean training with different cameras and lightening conditions, or even a wider range of users to make the model more robust.

Multi-Camera Systems:

Setting up multiple cameras could solve problems that are linked to limited angles and occlusions which could provide better input video to the model.

User Adaptability:

Future versions of the system, could include adaptivity features allowing the model to learn from different signing patterns. This can help in making the device become more generic or user friendly.

Real World Application:

The ultimate goal would be to get this model deployed in real world applications, for instance helping people with hearing impairments communicate or providing live translation services in public spaces.

Cross-Language Generalisation:

Another exciting area for future work is exploring the compatibility of sign languages

from different countries that could allow a universal sign language recognition system.

9.3. Final Thoughts

This is an important milestone, as such BSL recognition method was not yet possible before (at least to the best of our knowledge) through deep learning-based technologies. The current challenges are still far from eliminated, but this project has paved a way for further development. With digging deeper into the model, refining it and tackling the real-world challenges faced, an efficient sign language recognition could be developed. This will greatly benefit people who depend on sign language for communication.

To sum up, this project showcases not only the capabilities of current deep learning techniques but also teaches about the continuous need for reiteration and improvement when attempting to produce a technology which is useful in its true sense. The future of BSL recognition is undeniably bright and meaningful contributions to its promising horizon shall continue to be made.

References

Felman,A. (2024) *"ALS vs. Parkinson's disease: How do they differ?"* Available at: https://www.medicalnewstoday.com/articles/als-vs-parkinsons (Accessed: 18 April 2024)

Alyami, S. *et al*. (2024) 'Reviewing 25 years of continuous sign language recognition research: Advances, challenges, and prospects.' *Information Processing & Management*, *61*(5), article number 103774. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0306457324001341?via%3Dihub (Accessed: 2 September 2024)

Assan, M. and Grobel, K. (1998) " Video-based sign language recognition using Hidden Markov Models.", *In: Wachsmuth, I., Fröhlich, M. (eds) Gesture and Sign Language in Human-Computer Interaction*. GW 1997. Lecture Notes in Computer Science, vol 1371. Springer, Berlin, Heidelberg. Available at: https://link.springer.com/chapter/10.1007/BFb0052992 (Accessed: 13 June 2024)

Chong, T. and Lee, B. (2018) "American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach" , *Sensors,* 18(10), article number 3554. Available at: https://doi.org/10.3390/s18103554 (Accessed: 27 June 2024)

L. Zheng, B. Liang and A. Jiang. (2017) "Recent Advances of Deep Learning for Sign Language Recognition"' *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Sydney, NSW, Australia, 2017. Available at: https://ieeexplore.ieee.org/abstract/document/8227483 (Accessed: 09 August 2024)

Ma Y, Xu T, Kim K. (2022) "Two-Stream Mixed Convolutional Neural Network for American Sign Language Recognition.", *Sensors*. 22(16) article number 5959. Available at: https://doi.org/10.3390/s22165959 (Accessed: 21 July 2024)

Mienye ID, Swart TG, Obaido G. (2024) "Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications.", *Information*, 15(9), article number 517. Available at: https://doi.org/10.3390/info15090517 (Accessed: 31 August 2024)

Chakraboty, S., et al. (2023) “Sign Langugae Recognition Using Landmark Detection, GRU and LSTM”, *American Journal of Electronics & Communication*, 3(3), pp. 20-26(7). Available at: https://www.ingentaconnect.com/contentone/smart/ajec/2023/00000003/00000003/art00005 (Accessed: 29 July 2024)

Starner, T., Weaver, J. and Pentland, A. (1998) ‘Real-time American sign language recognition using desk and wearable computer based video,’ in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(20), pp. 1371-1375, , DOI: 10.1109/34.735811 (Accessed: 13 May 2024)

Molchanov, P., Gupta, S., Kim, K. & Kautz, J., (2015) ‘Hand gesture recognition with 3D convolutional neural networks.’ *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp.1-7. Available at: https://api.semanticscholar.org/CorpusID:206597741 (Accessed: 01 September 2024)

Sutton-Spence, R., & Woll, B. (1999). *The Linguistics of British Sign Language: An Introduction.* Cambridge University Press. Available at: https://books.google.co.uk/books?id=lUsCbazTICIC&pg=PA1&source=gbs_toc_r&cad=2#v=snippet&q=facial%20expressions&f=false (Accessed: 29 April 2024)

“British Greeting Signs” (no date) British Sign Language UK. Available at : https://www.british-sign.co.uk/bsl-greetings-signs-british-sign-language/ (Accessed: 20 May 2024)

Koller, O., Forster, J., & Ney, H. (2015) ‘Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers.’ *Computer Vision and Image Understanding,* 141, pp.108-125**.** Available at: https://www.sciencedirect.com/science/article/pii/S1077314215002088?via%3Dihub (Accessed: 19 August 2024)

Cahuantzi , R, Güttel, S & Chen, X (2023) ‘A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences.’ in *Intelligent Computing.* Available at: https://arxiv.org/pdf/2107.02248 (Accessed: 25 July 2024)