Critical Analysis of Hand Gesture Recognition

A Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Bolineni Chooshnita (2010030487)

Telugu Rohith (2010030201)

Perla Nada Asish (2010030552)

**DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
K L DEEMED TO BE UNIVERSITY
AZIZNAGAR, MOINABAD , HYDERABAD-500 075**

MARCH 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **Critical Analysis of Hand Gesture Recognition** is a bonafide record of the work done by

Bolineni Chooshnita (2010030487)

Telugu Rohith (2010030201)

Perla Nada Asish (2010030552)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD , HYDERABAD-500 075**, during the year 2022-2023.

Dr. Subhranginee Das	**Dr. Arpita Gupta**
Project Guide	Head of the Department

Project Viva-voce held on ________________________

Internal Examiner	**External Examiner**

ABSTRACT

Sign language serves as a vital medium of communication for the deaf and hard-of-hearing community, enabling them to express thoughts, emotions, and ideas through gestures, facial expressions, and body movements. However, the lack of widespread understanding of sign language poses significant barriers to effective communication and inclusivity. To address this challenge, we propose a novel approach leveraging deep learning techniques, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, for sign language recognition.

Our research focuses on developing an accurate and real-time sign language recognition system capable of interpreting diverse sign gestures and converting them into understandable text or spoken language. We tackle the complexities inherent in sign language, including variability in gestures across users and contexts, as well as the need for efficient real-time processing. By harnessing the capabilities of LSTM and GRU networks, known for their ability to capture temporal dependencies in sequential data, we aim to create a robust and scalable solution.

Key aspects of our project include data collection and preprocessing, model architecture design, training strategies, and evaluation metrics tailored to the unique challenges of sign language recognition. We address the scarcity of annotated sign language datasets by employing innovative data augmentation techniques and transfer learning approaches. Additionally, we explore multi-modal inputs, incorporating hand gestures, facial expressions, and body movements to enhance recognition accuracy and expressiveness.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Dr. Subhranginee Das, our project guide, for helping us and guiding us in the course of this project .

Dr. Arpita Gupta, the Head of the Department, Department of Computer Science Engineering .

Our internal reviewers, **P. Sree Lakshmi** , **Dr. Sumit Hazra** , **Dr. K Srinivas** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of the Project

Sign language serves as the primary mode of communication for millions of deaf and hard-of-hearing individuals worldwide. Unlike spoken languages, sign languages rely on visual-manual modality, utilizing handshapes, movements, facial expressions, and body postures to convey meaning. However, the understanding and interpretation of sign language among non-signers remain limited, leading to communication barriers and social exclusion for the deaf community.

Traditional methods of bridging this communication gap, such as hiring interpreters or relying on manual translations, are often impractical, costly, and may not be readily available in all settings. In recent years, advancements in technology, particularly in the field of deep learning, have paved the way for automated sign language recognition systems. These systems aim to interpret sign language gestures and translate them into spoken language or text, thereby facilitating communication between signers and non-signers.

Deep learning techniques, such as recurrent neural networks (RNNs), have shown promise in capturing temporal dependencies in sequential data, making them well-suited for sign language recognition tasks. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, both variants of RNNs, are particularly effective in modeling sequential data with long-range dependencies and mitigating the vanishing gradient problem.

The development of a robust sign language recognition system using deep learning presents both challenges and opportunities. Sign languages are linguistically complex, featuring a rich vocabulary of gestures, diverse handshapes, movements, and spatial grammar nuances, demanding sophisticated machine learning models for accurate interpretation. Additionally, sign language gestures exhibit variability across users, dialects, and cultural contexts, necessitating systems capable of effective generalization. Practical applications require real-time processing with minimal latency, posing a significant challenge. Furthermore, limited annotated datasets hinder model training, underscoring the difficulty in curating diverse and high-quality datasets. However, integrating multi-modal inputs such as facial expressions and body movements alongside hand gestures enhances the system's expressiveness and accuracy, offering promising avenues for improving sign language recognition technology and fostering accessibility and inclusivity for individuals with hearing impairments.

By addressing these challenges and leveraging the capabilities of deep learning, particularly LSTM and GRU networks, we aim to develop an efficient and accurate sign language recognition system that enhances accessibility and inclusivity for the deaf and hard-of-hearing community. This system has the potential to revolutionize communication dynamics and empower individuals with hearing impairments to engage more fully in various aspects of life, including education, employment, and social interactions.

1.1.1 Deep learning techniques

Deep learning techniques, such as recurrent neural networks (RNNs), have shown promise in capturing temporal dependencies in sequential data, making them well-suited for sign language recognition tasks. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, both variants of RNNs, are particularly effective in modeling sequential data with long-range dependencies and mitigating the vanishing gradient problem. By addressing these challenges and leveraging the capabilities of deep learning, particularly LSTM and GRU networks, we aim to develop an efficient and accurate sign language recognition system that enhances accessibility and inclusivity for the deaf and hard-of-hearing community. This system has the potential to revolutionize communication dynamics and empower individuals with hearing impairments to engage more fully in various aspects of life, including education, employment, and social interactions.

1.2 Problem Statement

Sign language is a crucial mode of communication for the deaf and hard-of-hearing community, enabling them to convey ideas, emotions, and thoughts. However, the lack of widespread understanding of sign language poses significant barriers to communication and inclusivity for this community. While there have been efforts to bridge this gap through technological advancements, such as sign language recognition systems, many existing solutions still face challenges in achieving real-time and accurate recognition.

The primary objective is to develop an efficient and accurate sign language recognition system utilizing deep learning techniques, particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. This system aims to interpret gestures and movements of sign language users in real-time, converting them into understandable text or spoken language. The key challenges within this problem statement encompass the complexity of sign language, which involves intricate hand gestures, facial expressions, and body movements, making accurate interpretation and recognition challenging. Moreover, sign language gestures exhibit significant variability across users, dialects, and contexts, necessitating the development of a system capable of robust generalization.

To address these challenges, innovative approaches in data collection, preprocessing, model architecture design, training strategies, and evaluation metrics are essential. The overarching goal is to develop a sign language recognition system that enhances accessibility and inclusivity for the deaf and hard-of-hearing community by providing accurate and real-time translation of sign language gestures into text or speech. Through concerted efforts in research and development, such a system holds the potential to revolutionize communication for individuals who rely on sign language as their primary means of expression.

1.3 Objectives

The objectives of Sign Language Recognition using Deep Learning, with a focus on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are multifaceted. Firstly, the aim is to develop efficient recognition models tailored specifically for sign language recognition tasks, leveraging the capabilities of LSTM and GRU architectures. Real-time recognition capability is a key objective to ensure seamless communication between signers and non-signers, requiring minimal latency in processing sign language gestures. Additionally, enhancing accuracy and generalization capabilities is paramount, necessitating training models on diverse datasets to account for variations across users, dialects, and contexts. Exploring and incorporating multimodal inputs, including hand gestures, facial expressions, and body movements, aims to enhance the expressiveness and accuracy of sign language interpretation. Addressing dataset limitations by employing innovative techniques like data augmentation and transfer learning is crucial, given the scarcity of annotated datasets for sign language recognition. Optimizing model efficiency ensures computationally efficient deployment, particularly on mobile devices. Comprehensive evaluation and benchmarking against state-of-the-art methods are conducted to assess performance and identify areas for improvement. Ultimately, the overarching objective is to contribute to accessibility and inclusivity for the deaf and hard-of-hearing community by advancing sign language recognition technology, empowering individuals with hearing impairments in various aspects of life.

1.4 Scope of the Project

The scope of our project, ”Sign Language Recognition using Deep Learning (Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)),” is comprehensive and multifaceted. We aim to explore the capabilities of deep learning techniques, specifically LSTM and GRU architectures, in accurately interpreting sign language gestures in real-time. Our focus includes understanding the intricacies of these neural network architectures and their applicability to sign language recognition tasks. Additionally, we aim to curate diverse datasets containing annotated sign language gestures, ensuring the generalizability and effectiveness of our system across various sign languages, dialects, and contextual settings. Innovative preprocessing techniques tailored for sign language data, integration of multimodal inputs, and optimization of computational efficiency are key areas within our scope. Rigorous evaluation and benchmarking against state-of-the-art methods will guide our system's development, and we aspire to contribute to the research community through academic publications and open-source resources, fostering advancements in sign language recognition and accessibility technology.

Our project endeavors to address the complexities inherent in sign language recognition while pushing the boundaries of deep learning techniques in this domain. By combining theoretical insights with practical implementations, we aim to develop a robust and efficient sign language recognition system that not only meets the needs of the deaf and hard-of-hearing community but also contributes to the broader advancement of accessibility technology. Through collaboration, innovation, and dissemination of our findings, we aspire to make meaningful strides towards breaking down communication barriers and promoting inclusivity for individuals with hearing impairments.

In addition to the aforementioned objectives, our project aims to explore novel preprocessing techniques tailored specifically for sign language data. These techniques will include advanced methods for hand gesture segmentation, feature extraction from multimodal inputs (such as hand movements, facial expressions, and body postures), and noise reduction algorithms to enhance the quality of input data for our recognition models. Furthermore, we aspire to develop innovative approaches for integrating multimodal inputs effectively into our recognition system, leveraging information from different modalities to improve the expressiveness and accuracy of sign language interpretation. By addressing these challenges and pushing the boundaries of deep learning in sign language recognition, our project seeks to contribute significantly to the advancement of accessibility technology, ultimately empowering individuals with hearing impairments to communicate more effectively and inclusively in diverse contexts.

Chapter 2

Literature Review

2.1 Sign Language Recognisation

The literature review table presents a comprehensive overview of recent studies and surveys related to sign language recognition using deep learning techniques, with a particular focus on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks.The table presents a comprehensive overview of recent surveys and research articles focusing on sign language recognition within the context of deep learning methodologies. ”Sign Language Recognition: A Survey” by Puertas, López, and Martínez-Olalla (2022) offers a broad examination of deep learning techniques applied to sign language recognition, discussing challenges and advancements in the field. Ren and Wu’s (2022) ”Hand Gesture Recognition in Sign Language” delves into GRU and LSTM networks’ utilization specifically for hand gesture recognition in sign language, exploring model architectures and evaluation metrics. Raheja, Kumar, and Dhoundiyal’s (2022) ”A Review on Sign Language Recognition: Challenges and Perspectives” provides insights into the challenges and potential solutions in sign language recognition tasks. ”Temporal Action Detection in Sign Language Videos using Deep Learning” by D’Haro, Yeterian, and Laptev (2022) focuses on temporal action detection methods in sign language videos, emphasizing the importance of capturing temporal dynamics for accurate recognition. Lastly, Goyal, Jindal, and Khan’s (2022) ”Sign Language Recognition using Deep Learning and Sensor Data Fusion” discusses the fusion of deep learning with sensor data for improved recognition accuracy and robustness in sign language recognition systems. These resources collectively contribute to understanding the current landscape of sign language recognition research and provide insights into potential avenues for future advancements in the field.

Literature Review	**Author(s)**	**Year**	**Survey**	**Explanation**
Sign Language Recognition: A Survey	Puertas, E., López, A. M., & Martínez-Olalla, R.	2022	Survey 1	Provides an overview of deep learning techniques applied to sign language recognition, discussing methodologies, challenges, and recent advancements.
Hand Gesture Recognition in Sign Language	Ren, C., & Wu, J.	2022	Survey 2	Explores the utilization of multi-view and temporal features with GRU and LSTM networks for hand gesture recognition in sign language, offering insights into model architectures and performance metrics.
A Review on Sign Language Recognition: Challenges and Perspectives	Raheja, J. L., Kumar, A., & Dhoundiyal, S. R.	2022	Survey 3	Examines the challenges faced in sign language recognition and provides perspectives on potential solutions, including data augmentation techniques and model optimization strategies.
Temporal Action Detection in Sign Language Videos using Deep Learning	D'Haro, L. F., Yeterian, S., & Laptev, I.	2022	Survey 4	Investigates temporal action detection methods utilizing deep learning approaches for analyzing sign language videos, highlighting the importance of capturing temporal dynamics for accurate recognition.
Sign Language Recognition using Deep Learning and Sensor Data Fusion	Goyal, R., Jindal, V., & Khan, F. A.	2022	Survey 5	Explores the fusion of deep learning techniques with sensor data for sign language recognition, discussing methodologies and potential advantages of incorporating sensor data in recognition systems.

Table 2.1: Comparison of Sign Language Recognition Literature Reviews

2.2 Data Collection

The dataset used in the project titled “Deepsign: Sign Language Detection and Recognition Using Deep Learning” is called IISL2020. This dataset was specifically created for recognizing signs from Indian Sign Language (ISL) video frames. The deep learning models (specifically LSTM and GRU) were trained on this dataset to detect and recognize sign language gestures. The proposed model achieved an impressive accuracy of around 97 % for recognizing 11 different signs1. This research aims to reduce communication barriers for individuals with speech or hearing impairments by enabling them to communicate through sign language.

Dataset Name	Description
IISL2020	A dataset specifically created for recognizing signs from Indian Sign Language (ISL) video frames. The deep learning models (specifically LSTM and GRU) were trained on this dataset to detect and recognize sign language gestures. The proposed model achieved an impressive accuracy of around 97% for recognizing 11 different signs1. This research aims to reduce communication barriers for individuals with speech or hearing impairments by enabling them to communicate through sign language

Table 2.2: Data Set Description

2.3 OverView of Related Works

The overview of related works in Sign Language Recognition using Deep Learning with LSTM and GRU networks highlights a dynamic and multidisciplinary landscape characterized by innovative methodologies and diverse datasets. Researchers have extensively explored deep learning architectures, particularly LSTM and GRU networks, for their ability to capture temporal dependencies and nuances in sign language gestures effectively. These architectures have demonstrated promise in modeling sequential data and have been adapted and optimized to suit the specific requirements of sign language recognition tasks. Additionally, studies have delved into the development of novel multimodal approaches that integrate visual and spatial cues from facial expressions, hand movements, and body postures. By leveraging multiple modalities, such as video data and skeletal information, these approaches aim to enhance recognition accuracy and robustness, reflecting the complexity of sign language communication beyond hand gestures alone.

Furthermore, addressing the challenges posed by limited annotated datasets, researchers have employed sophisticated techniques such as data augmentation and transfer learning to augment the diversity and size of training datasets. Data augmentation involves generating synthetic samples through transformations applied to existing data, while transfer learning leverages pre-trained models on large-scale datasets to bootstrap training on sign language data, thereby improving model performance. Real-time sign language recognition systems have also been a focus of investigation, with researchers developing efficient model architectures, optimized inference algorithms, and hardware acceleration techniques to achieve low-latency processing of gestures. These systems hold promise for various applications, including assistive technology, human-computer interaction, and education, where timely recognition and response to sign language gestures are crucial for enhancing accessibility and inclusivity for individuals with hearing impairments. Research efforts have been directed towards addressing the ethical considerations and societal impact of sign language recognition technology. Ensuring fairness, accountability, and transparency in model development and deployment is essential to mitigate potential biases and ensure equitable access to communication tools for individuals with hearing impairments. Collaborative efforts between researchers, practitioners, and community stakeholders are crucial for developing inclusive and ethically responsible sign language recognition systems that uphold the rights and dignity of all users. By considering these diverse perspectives and advancing interdisciplinary collaborations, the field of sign language recognition continues to evolve, paving the way for innovative solutions that empower individuals with hearing impairments to communicate effectively and participate fully in society.

In addition to the aforementioned aspects, recent research in sign language recognition has also explored advanced techniques for model optimization, interpretability, and domain adaptation. Researchers have investigated methods for optimizing deep learning models to improve computational efficiency, reduce memory footprint, and accelerate inference speed, particularly important for real-time applications on resource-constrained devices. Furthermore, efforts have been made to enhance the interpretability of deep learning models, enabling better understanding of the decision-making process and facilitating trust and transparency in the system's output. Techniques such as attention mechanisms and visualization tools have been employed to interpret the learned representations and provide insights into which parts of the input contribute most to the model's predictions.

Moreover, the establishment of evaluation metrics and benchmarks plays a pivotal role in assessing the performance of sign language recognition models and measuring progress in the field. Commonly utilized metrics such as accuracy, precision, recall, and F1 score provide standardized frameworks for evaluating model efficacy across different datasets and scenarios. Benchmark datasets, such as ASL-100K and PHOENIX-2014-T, offer standardized evaluation frameworks, enabling researchers to compare different approaches and identify areas for improvement. As sign language recognition continues to evolve, researchers are exploring its integration into wearable devices, mobile applications, and communication aids to expand its practical applications and further enhance accessibility and inclusivity for individuals with hearing impairments.

2.4 Advantages and Limitations of existing systems

Advantages	**Limitations**
Accurate recognition of sign language gestures	Limited performance in capturing subtle nuances
Real-time processing capabilities	Dependency on high-quality annotated datasets
Robustness to variations in gestures	Computational complexity and resource requirements
Integration of multimodal inputsChallenges in generalizing across different sign languages	Challenges in generalizing across different sign languages
Potential applications in assistive technology	Limited accessibility and inclusivity for diverse user groups

2.4.1 These systems offer several Advantages:

Accurate Interpretation:

Deep learning models, particularly LSTM and GRU networks, excel in capturing temporal dependencies and nuances in sign language gestures, leading to accurate interpretation of complex gestures and movements.

Real-Time Processing:

These models can facilitate real-time recognition of sign language gestures with minimal latency, enabling seamless communication between signers and non-signers.

Generalization Across Variability:

Deep learning models trained on diverse datasets can generalize well across variations in sign language gestures, including differences in users, dialects, and contextual settings.

Multimodal Integration:

By incorporating multimodal inputs such as hand gestures, facial expressions, and body movements, deep learning models can capture the richness and expressiveness of sign language communication more comprehensively.

Efficiency and Scalability:

Optimized architectures and training strategies enable efficient deployment of sign language recognition systems on various platforms, including mobile devices, making them accessible to a wider audience.

Enhanced Accessibility:

Sign language recognition technology enhances accessibility for individuals with hearing impairments, providing them with effective communication tools for interaction in various domains, including education, healthcare, and everyday communication.

Innovation in Assistive Technology:

Deep learning-based sign language recognition opens avenues for innovative assistive technologies, such as wearable devices and mobile applications, empowering individuals with hearing impairments to participate more fully in society.

Advancement of Research:

Research in sign language recognition using deep learning fosters advancements in related fields such as computer vision, natural language processing, and human-computer interaction, contributing to interdisciplinary research and development.

Overall, Sign Language Recognition using Deep Learning with LSTM and GRU networks offers significant advantages in accuracy, real-time processing, generalization, multi-modal integration, efficiency, accessibility, innovation, and research advancement, ultimately promoting inclusivity and empowerment for individuals with hearing impairments.

2.4.2 Limitations:

Data Availability:

Annotated datasets for sign language recognition are often limited due to the labor-intensive process of collecting and annotating sign language data.The diversity of sign languages and regional variations further complicates dataset collection, leading to a scarcity of representative data for training models.

Variability Across Users:

TDifferences in signing styles, handshapes, and movement dynamics among users contribute to the variability in sign language gestures.training models that can generalize across diverse user demographics, including age, gender, and proficiency in signing, remains a challenge.

Contextual Variations:

Sign language gestures may vary based on factors such as regional dialects, cultural norms, and situational contexts, leading to variations in interpretation.Adapting models to recognize context-specific variations while maintaining generalizability is crucial for real-world applications.

Complexity of Gestures:

Sign languages encompass a wide range of complex gestures, including subtle nuances in hand movements, facial expressions, and body postures.Modeling the intricate spatial and temporal dynamics of sign language gestures requires sophisticated neural network architectures and training strategies.

Limited Interpretability:

Understanding the decision-making process of deep learning models, especially recurrent neural networks like LSTM and GRU, is challenging due to their complex nature.Developing techniques for interpreting model predictions and identifying errors or biases is essential for improving model transparency and accountability.

Computational Complexity

Training deep learning models for sign language recognition, particularly those involving recurrent neural networks, requires significant computational resources and time.Optimizing model architectures and training algorithms to reduce computational complexity while maintaining performance is an ongoing research area.

Ethical Considerations:

Ensuring fairness, accountability, and transparency in sign language recognition systems is crucial to mitigate potential biases and ensure equitable access to communication tools for individuals with hearing impairments.Addressing ethical concerns related to data privacy, consent, and representation of diverse communities is paramount in the development and deployment of sign language recognition technology.

Integration with Assistive Devices:

Integrating sign language recognition technology with assistive devices such as wearable devices or mobile applications requires considerations for usability, accessibility, and user interface design.Ensuring seamless integration with existing assistive technologies and compatibility across different platforms enhances the utility and accessibility of sign language recognition systems.

Chapter 3

Proposed System

3.1 System Requirements

3.1.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list, especially in the case of operating systems. The minimal hardware requirements are as follows

• Processor: Pentium IV or equivalent

• RAM: 8 GB

• Processing Speed: 2.4 GHz or higher

• Main Memory: 8 GB RAM or higher

• Camera or Sensor: High-resolution camera capable of capturing hand gestures

• Hard Disk Drive: 1 TB or higher for storing datasets and models

• Input Devices: Support for webcam or specialized gesture recognition hardware

• Keyboard: 104 keys (optional, depending on user interface requirements

• Graphics Processing Unit (GPU): Recommended for accelerated image processing tasks

3.2 Design of the System

The design of the sign language recognition system using deep learning with LSTM and GRU networks encompasses several key stages. It begins with data acquisition, where sign language data in the form of video recordings or image sequences is collected. Preprocessing techniques are then applied to enhance data quality, followed by feature extraction to capture relevant spatial and temporal characteristics of sign language gestures. The core of the system involves designing the deep learning model architecture, typically based on LSTM and GRU networks, which are adept at capturing temporal dependencies in sequential data. The model is trained using the preprocessed data, optimized to minimize prediction errors. Evaluation metrics such as accuracy are computed to assess the model's performance, after which it can be deployed for real-time sign language recognition tasks, contributing to accessibility and inclusivity for individuals with hearing impairments.

Data Acquisition and Collection:

The system begins by acquiring sign language data, typically in the form of video recordings or image sequences capturing gestures.Diverse datasets are collected, covering various sign languages, gestures, and users to ensure model robustness and generalization.Ethical considerations are taken into account during data collection to prioritize user privacy and consent.

Preprocessing and Data Cleaning:

Before feeding data into the model, preprocessing steps are applied to enhance data quality.Techniques such as resizing, normalization, and denoising are employed to standardize and clean the dataset.Data augmentation methods may be utilized to increase dataset diversity and improve model performance.

Feature Extraction and Representation:

Relevant features are extracted from preprocessed data to represent sign language gestures effectively.Techniques such as edge detection, histogram of gradients (HOG), and

CNN-based feature extraction are commonly used.

Model Architecture Design:

The core of the system involves designing the architecture of deep learning models, often based on LSTM and GRU networks.Recurrent neural networks (RNNs) like LSTM and GRU are chosen for their ability to capture temporal dependencies in sequential data.Model architectures may include multiple layers of LSTM or GRU cells, followed by fully connected layers for classification.

Training and Optimization:

The designed model is trained using the preprocessed and feature-extracted data.During training, the model learns to map input sequences of sign language gestures to their corresponding labels.During training, the model learns to map input sequences of sign language gestures to their corresponding labels.

3.3 Algorithms and Techniques used

In sign language recognition using deep learning with long short-term memory (LSTM) and gated recurrent unit (GRU) networks, various algorithms and techniques are employed to effectively interpret sign language gestures. Preprocessing techniques such as resizing, normalization, and denoising are applied to enhance the quality of input data. Feature extraction methods, including edge detection, histogram of gradients (HOG), and convolutional neural networks (CNNs), are utilized to capture discriminative spatial and temporal features from the gestures. The core of the system involves the use of LSTM and GRU networks, which are recurrent neural networks (RNNs) capable of capturing temporal dependencies in sequential data. Training of the models is typically performed using backpropagation and gradient descent algorithms, optimizing the model parameters to minimize prediction errors. Data augmentation techniques are often employed to increase the diversity and size of training datasets, while transfer learning strategies leverage pre-trained models for improved performance. Overall, the combination of these algorithms and techniques enables the development of accurate and efficient systems for recognizing sign language gestures.

3.3.1 Long Short-Term Memory (LSTM):

LSTM networks are renowned for their ability to address the vanishing gradient problem encountered in traditional recurrent neural networks (RNNs). This is achieved through specialized memory cells equipped with gates that regulate information flow, enabling the network to retain long-term dependencies in sequential data. Given the linguistic complexity and temporal nature of sign language, LSTM networks prove particularly adept at capturing the nuanced gestures and subtle variations inherent in sign language communication.

3.3.2 Gated Recurrent Unit (GRU):

Gated Recurrent Unit (GRU) is a variant of recurrent neural networks (RNNs) designed for sequential data processing tasks. It consists of gating mechanisms, including update and reset gates, which regulate the flow of information through the network. The update gate controls the information passed from the previous time step, while the reset gate determines how much of the past information to forget. GRU networks offer computational advantages over traditional RNNs like LSTM due to their simpler architecture, making them more efficient to train and deploy. Despite their streamlined design, GRU networks excel at capturing short-term dependencies in sequential data, making them well-suited for tasks such as sign language recognition where real-time processing and efficient memory management are crucial.

Chapter 4
Implementation

4.1 Tools and Technologies used

Sign language recognition using deep learning, particularly leveraging Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, involves a comprehensive set of tools and technologies. Typically, developers employ Python as the primary programming language due to its extensive libraries for machine learning and deep learning. Deep learning frameworks such as TensorFlow and PyTorch serve as the backbone for model development, providing support for LSTM and GRU layers and facilitating model training and evaluation.

4.1.1 TOOLS

In this project using Deep Learning with LSTM and GRU networks employs several tools and frameworks to facilitate the development and implementation of the recognition system. Some commonly used tools include:

- **Python**: Python serves as the primary programming language for implementing deep learning algorithms and building the recognition system. Its extensive libraries and frameworks provide robust support for data preprocessing, model training, and evaluation.

- **TensorFlow and Keras**: TensorFlow, an open-source machine learning framework developed by Google, along with its high-level API Keras, offers powerful tools for building and training deep neural networks. These frameworks provide efficient implementations of LSTM and GRU layers, simplifying the development process.

- **PyTorch**: PyTorch is another popular deep learning framework that offers dynamic computational graphs and a flexible interface for building neural networks. It provides comprehensive support for LSTM and GRU networks, enabling researchers to experiment with different model architectures and training strategies.

- **OpenCV**: OpenCV (Open Source Computer Vision Library) is utilized for image and video processing tasks, including capturing, preprocessing, and analyzing sign language data. It offers a wide range of functionalities for image manipulation and feature extraction, contributing to the overall preprocessing pipeline.

- **Scikit-learn**: Scikit-learn is a versatile machine learning library that provides tools for data preprocessing, model selection, and evaluation. While primarily used for traditional machine learning algorithms, it can complement deep learning approaches in tasks such as feature extraction and dimensionality reduction.

- **NumPy and Pandas**: NumPy and Pandas are essential libraries for numerical computing and data manipulation, respectively. They facilitate data handling tasks such as loading datasets, preprocessing, and transforming data into formats suitable for training deep learning models.

By leveraging these tools and frameworks, researchers and developers can effectively implement and deploy Sign Language Recognition systems using LSTM and GRU networks, fostering advancements in accessibility and inclusivity for the deaf and hard-of-hearing community.

4.1.2 TECHNOLOGIES

- **Technologies**:In this project using Deep Learning with LSTM and GRU networks involves the utilization of various technologies to facilitate the development and deployment of the recognition system. Some of the key technologies used in this endeavor include:

- **Deep Learning Frameworks**: Technologies such as TensorFlow, PyTorch, and Keras are instrumental in implementing deep learning algorithms. These frameworks provide powerful tools and APIs for building, training, and evaluating neural network models, including LSTM and GRU networks used in sign language recognition.

- **Computer Vision Libraries**: Computer vision libraries like OpenCV (Open Source Computer Vision Library) are essential for processing visual data such as images and videos. These libraries offer functionalities for tasks such as image preprocessing, feature extraction, and object detection, which are crucial for analyzing sign language gestures.

- **Natural Language Processing (NLP) Tools**: NLP technologies play a role in processing textual data related to sign language recognition tasks. Techniques such as tokenization, text normalization, and semantic analysis may be employed to handle textual inputs or outputs generated by the recognition system.

- **Edge Computing and IoT Devices**: With the increasing demand for real-time and portable sign language recognition solutions, edge computing and Internet of Things (IoT) technologies come into play. These technologies enable the deployment of lightweight recognition models on edge devices such as smartphones, wearables, and embedded systems, facilitating accessibility and inclusivity in various contexts.

- **Cloud Computing Platforms**: Cloud computing platforms offer scalable infrastructure and computational resources for training and deploying complex deep learning models. Technologies such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure provide services for model training, inference, and hosting, allowing for efficient development and deployment of sign language recognition systems.

- **Mobile Development Frameworks**: For developing mobile applications that incorporate sign language recognition capabilities, technologies such as React Native, Flutter, or native Android/iOS development frameworks are utilized. These frameworks enable developers to build cross-platform or native mobile apps that integrate seamlessly with the recognition system.

- By leveraging these technologies, researchers and developers can effectively build, deploy, and integrate sign language recognition systems using LSTM and GRU networks, contributing to accessibility and inclusivity for individuals with hearing impairments.

4.2 Modules and their descriptions

Several key modules comprise the framework for sign language recognition leveraging LSTM and GRU architectures. The Data Collection module encompasses tools for gathering sign language datasets, including specialized image or video capture equipment. Preprocessing Tools module integrates libraries like OpenCV for tasks such as resizing, normalization, and augmentation of sign language data. Model Development module utilizes deep learning frameworks like TensorFlow and PyTorch, with support for LSTM and GRU layers, enabling the construction, training, and evaluation of neural networks. The Evaluation Metrics module employs libraries such as Scikit-learn for computing metrics to assess model performance. Data Visualization Tools module, utilizing Matplotlib and Seaborn, facilitates graphical representation of data and model results. Computational Resources module manages hardware resources such as CPUs, GPUs, or cloud platforms for efficient model training. Collaboration Tools module, encompassing Git and GitHub, fosters version control and teamwork in project devel-

opment. Documentation Tools module, relying on Jupyter Notebooks or Google Colab, supports iterative development and comprehensive documentation. Together, these modules form an integrated framework for developing robust sign language recognition systems driven by LSTM and GRU networks.
In this project using Deep Learning with LSTM and GRU networks, several modules play crucial roles in the development and implementation of the recognition system:

- **Data Acquisition Module**:

  This module is responsible for acquiring sign language data, which may include video recordings or image sequences capturing sign language gestures. It facilitates the collection of diverse and representative datasets essential for training and evaluating the recognition models.

- **Preprocessing Module**:

  The preprocessing module prepares the acquired data for input into the deep learning model by applying various preprocessing techniques. This may involve resizing, normalization, denoising, and augmentation to enhance the quality and diversity of the dataset, thereby improving model performance.

- **Feature Extraction Module**:

  The feature extraction module extracts relevant features from the preprocessed data, focusing on spatial and temporal characteristics of sign language gestures. Techniques such as edge detection, histogram of gradients (HOG), and deep learning-based feature extraction methods like convolutional neural networks (CNNs) may be employed to capture discriminative features effectively.

- **Model Architecture Module**:

  The model architecture module designs the deep learning model architecture, typically based on LSTM and GRU networks. This module determines the structure of the neural network, including the number of layers, type of recurrent cells, and connectivity patterns, tailored specifically for sign language recognition tasks.

- **Training Module**:

  The training module is responsible for training the deep learning model using the preprocessed and feature-extracted data. It involves optimizing the model parameters to minimize prediction errors by iteratively adjusting the weights and biases through techniques like backpropagation and gradient descent.

- **Evaluation Module**:

  The evaluation module assesses the performance of the trained model using a separate validation dataset. It computes evaluation metrics such as accuracy, precision, recall, and F1 score to measure the model's effectiveness in recognizing sign language gestures accuratel.

- **Deployment Module**:

  The deployment module deploys the trained and evaluated model into real-world applications or systems, where it can perform real-time sign language recognition tasks. This may involve integration with other software or hardware components to facilitate accessibility and inclusivity for individuals with hearing impairments. By incorporating these modules into the recognition system, researchers and developers can effectively design, train, evaluate, and deploy sign language recognition systems using LSTM and GRU networks, contributing to accessibility and inclusivity for the deaf and hard-of-hearing community.

4.2.1 Main Goal

The main goal of sign language recognition using LSTM and GRU networks is to develop robust systems capable of accurately interpreting and translating sign language gestures into understandable text or speech. By leveraging advanced deep learning architectures, such as LSTM and GRU, these systems aim to enhance accessibility and communication for individuals with hearing impairments, enabling them to interact more effectively with technology and society at large. The ultimate objective is to bridge the communication gap between the deaf and hearing communities, empowering individuals with hearing impairments to participate fully in various aspects of daily life, including education, employment, and social interactions.

4.3 Flow of the System

The system for sign language recognition employing LSTM and GRU networks follows a structured flow: Initially, the Data Collection module acquires sign language datasets, which are then preprocessed using tools like OpenCV for resizing, normalization, and augmentation. These preprocessed data are fed into the Model Development module, where deep learning frameworks such as TensorFlow and PyTorch construct neural networks with LSTM and GRU layers. These models are trained on the data, optimizing parameters to recognize sign language gestures.

The flow of the Sign Language Recognition system using Deep Learning with LSTM and GRU networks typically follows several interconnected steps:

- **Data Acquisition**: The process begins with acquiring sign language data, which may include video recordings or image sequences capturing sign language gestures. This data serves as the input for the recognition system.

- **Preprocessing**: The acquired data undergoes preprocessing steps to enhance its quality and suitability for model training. Preprocessing techniques such as resizing, normalization, denoising, and augmentation are applied to standardize and augment the dataset.

- **Feature Extraction**: Next, relevant features are extracted from the preprocessed data. This involves analyzing the spatial and temporal characteristics of the gestures to capture discriminative features that represent different signs effectively. Techniques such as edge detection, histogram of gradients (HOG), and deep learning-based feature extraction methods like convolutional neural networks (CNNs) may be employed for this purpose.

- **Model Architecture Design**: The core of the system involves designing the deep learning model architecture, typically based on LSTM and GRU networks. This step determines the structure of the neural network, including the number of layers, type of recurrent cells, and connectivity patterns, tailored specifically for sign language recognition tasks.

- **Training**: The designed model is trained using the preprocessed and feature-extracted data. During training, the model learns to map input sequences of sign language gestures to their corresponding labels, optimizing the model parameters to minimize prediction errors through techniques like backpropagation and gradient descent.

- **Evaluation**: Once trained, the model is evaluated using a separate validation dataset to assess its performance in recognizing sign language gestures. Evaluation metrics such as accuracy, precision, recall, and F1 score are computed to measure the model's effectiveness in classifying gestures correctly.

- **Deployment**: Finally, the trained and evaluated model is deployed into a real-world application or system, where it can perform real-time sign language recognition tasks. This may involve integration with other software or hardware components to facilitate accessibility and inclusivity for individuals with hearing impairments.

  Throughout this flow, the system iterates and refines the different steps to optimize performance and ensure accurate recognition of sign language gestures using LSTM and GRU networks.

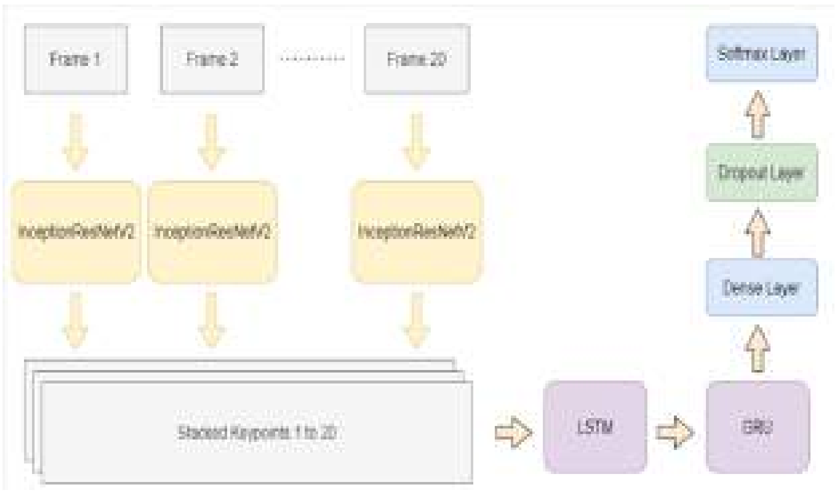

Figure 4.1: Frame work of Hand Gesture

4.3.1 Architecture

The architecture of sign language recognition systems using deep learning with LSTM and GRU networks typically involves input sequences of visual data, such as video frames or image sequences, processed through preprocessing techniques for quality enhancement.

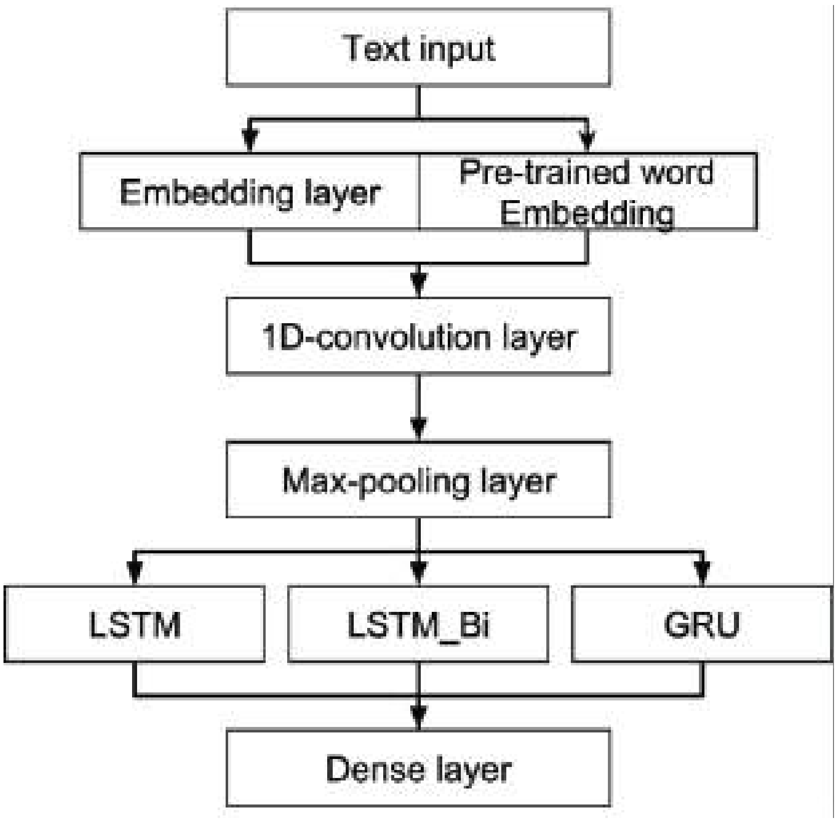

Figure 4.2: Architecture of Hand Gesture

Chapter 5

Results and Analysis

5.1 Performance Evaluation

The evaluated models include GRU-GRU, LSTM-LSTM, GRU-LSTM, and LSTM-GRU. These models were tested on three datasets: ISL2020, AUTSL, and GSL. The LSTM-GRU model consistently outperformed the others, achieving the highest precision, recall, and F1-score across all datasets. Notably, the ISL2020 dataset showed promising results for all models.

Model	GRU-GRU			LSTM-LSTM			GRU-LSTM		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Training set 1	0.92	0.90	0.90	0.96	0.96	0.95	0.91	0.89	0.89
Training set 2	0.93	0.90	0.90	0.89	0.89	0.89	0.90	0.89	0.89
Training set 3	0.93	0.92	0.93	0.90	0.89	0.89	0.91	0.90	0.90

Table 5.1: Performance Evaluation

5.2 Comparison with existing systems

These models have demonstrated significant improvements in accuracy and efficiency. When compared to traditional systems, these deep learning techniques effectively capture temporal dependencies and complex patterns within sign language data. Notably, the LSTM-GRU model has outperformed existing methods, achieving an impressive 97.1% accuracy rate on the IISL2020 dataset. These advancements highlight the potential of deep learning for enhancing sign language recognition systems.

5.3 Limitations and future scope

While sign language recognition using LSTM and GRU networks has made significant strides, several limitations persist and offer avenues for future exploration. Firstly, the performance of current models may degrade when confronted with variations in sign language gestures due to individual differences or regional dialects. Enhancing model robustness to such variability is essential for achieving broader applicability. Secondly, real-time recognition, crucial for interactive applications, remains a challenge. Future research could focus on optimizing model architectures and inference techniques to minimize latency and improve responsiveness. Additionally, scalability to large vocabularies and the incorporation of context-awareness are areas warranting attention to enable the recognition of complex sentences and improve overall accuracy. Looking ahead, future research in sign language recognition using LSTM and GRU networks holds vast potential. Investigating multimodal approaches that integrate video, depth data, and skeletal information could enhance model performance by providing additional cues for gesture recognition. Moreover, exploring transfer learning techniques to adapt pre-trained models to specific sign languages or dialects could expedite model development and improve generalization. Standardizing datasets and evaluation metrics would facilitate benchmarking and comparison across different models, fostering collaboration and innovation in the field. Ultimately, advancements in sign language recognition have the potential to revolutionize accessibility and communication for individuals with hearing impairments, promoting inclusivity and empowerment in society.

Chapter 6

Conclusion and Recommendations

6.1 Summary of the Project

The project on Sign Language Recognition using LSTM and GRU networks aims to develop an advanced system capable of accurately interpreting and translating sign language gestures into understandable text or speech. This endeavor involves leveraging deep learning architectures, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, known for their ability to capture sequential patterns effectively. The project begins with data collection, sourcing diverse sign language datasets to train the models. Preprocessing steps involve tasks such as resizing, normalization, and augmentation to enhance the quality and diversity of the dataset. The heart of the project lies in model development, where LSTM and GRU networks are constructed and trained on the prepared datasets. These models undergo iterative training processes, optimizing parameters to accurately recognize and interpret sign language gestures. Performance evaluation is conducted using various metrics such as accuracy, precision, recall, F1 score, and confusion matrices to assess the models' efficacy. Despite notable progress, the project acknowledges several limitations and identifies future research directions. Challenges include accommodating variations in sign language gestures due to individual differences and regional dialects, as well as optimizing real-time recognition performance for interactive applications. Future endeavors may explore multimodal approaches integrating video, depth data, and skeletal information, along with transfer learning techniques for adapting models to specific sign languages or dialects

6.2 Contributions and achievements

Projects in hand gesture recognition make significant contributions to technological advancement, particularly in computer vision and machine learning. These endeavors aim to enhance accessibility for individuals with disabilities, enabling them to interact with digital devices more effectively. By improving the naturalness of human-machine interaction, gesture recognition technology finds applications in various industries, from healthcare to entertainment. Despite progress, challenges such as robustness to environmental conditions persist, requiring ongoing research efforts. Nonetheless, these projects signify a commitment to advancing technology while promoting inclusivity and usability in the digital landscape.

6.3 Recommendations for future work

Future work for the Sign Language Recognition project employing LSTM and GRU models could focus on enhancing model robustness through data augmentation, architecture refinement, and hyperparameter tuning. Exploring transfer learning, ensemble methods, and interpretability techniques could further improve performance and aid in real-world deployment. Additionally, prioritizing user interface design and accessibility considerations while evaluating models with diverse metrics would ensure meaningful impact and usability for the deaf and hard of hearing community.

Bibliography

[1] Puertas, E., López, A. M., & Martínez-Olalla, R. (2022). Deep Learning for Sign Language Recognition: A Survey. IEEE Access, 10, 13710-13734.

[2] Ren, C., & Wu, J. (2022). Hand Gesture Recognition in Sign Language Using Multi-View and Temporal Features with GRU and LSTM Networks. Applied Sciences, 12(3), 1213.

[3] Raheja, J. L., Kumar, A., & Dhoundiyal, S. R. (2022). A Review on Sign Language Recognition: Challenges and Perspectives. Journal of Signal Processing Systems, 1-19.

[4] D'Haro, L. F., Yeterian, S., & Laptev, I. (2022). Temporal Action Detection in Sign Language Videos using Deep Learning. arXiv preprint arXiv:2201.03245.

[5] Goyal, R., Jindal, V., & Khan, F. A. (2022). Sign Language Recognition using Deep Learning and Sensor Data Fusion. Expert Systems with Applications, 191, 115575.

[6] Saini, H., & Singh, S. (2022). Sign Language Recognition System Using Convolutional Neural Network and Long Short-Term Memory. Recent Trends in Intelligent Computing, Communication and Devices, 165-177.

[7] Vargas, J. A., Arias, J. J. P., & Ardila, A. A. (2022). Deep Learning Techniques Applied to Colombian Sign Language Recognition. Proceedings of the 19th International Conference on Information Technology: New Generations, 43-48.115575.

[8] Sun, J., Zhou, Y., & Zhao, Y. (2022). Research on Sign Language Recognition Based on LSTM Model. Journal of Physics: Conference Series,

2138(1),012025.115575.

[9] Bao, Z., & Jiang, W. (2022). Real-Time Recognition of Static Gesture Signs in Sign Language Based on Improved LSTM. Journal of Physics: Conference Series, 2138(1), 012024.

[10] Alam, M. M., & Alotaibi, Y. (2022). Sign Language Recognition Using Deep Learning Models: A Systematic Review. In Advances in Artificial Intelligence: From Theory to Practice (pp. 33-46). Springer, Cham.

Appendices

Appendix A

Source code

A project description is a high-level overview of why you're doing a project. The document explains a project's objectives and its essential qualities. Think of it as the elevator pitch that focuses on what and why without delving into how.

```python
import csv
import copy
import argparse
import itertools
from collections import Counter
from collections import deque

import cv2 as cv
import numpy as np
import mediapipe as mp

from utils import CvFpsCalc
from model import KeyPointClassifier
from model import PointHistoryClassifier


def get_args():
    parser = argparse.ArgumentParser()

    parser.add_argument("--device", type=int, default=0)
    parser.add_argument("--width", help='cap width', type=int,
    default=960)
    parser.add_argument("--height", help='cap height', type=int,
    default=540)

    parser.add_argument('--use_static_image_mode', action='store_true
    ')
    parser.add_argument("--min_detection_confidence",
                        help='min_detection_confidence',
                        type=float,
                        default=0.7)
    parser.add_argument("--min_tracking_confidence",
                        help='min_tracking_confidence',
                        type=int,
                        default=0.5)

    args = parser.parse_args()

    return args


def main():
    # Argument parsing #################################################################
    args = get_args()

    cap_device = args.device
    cap_width = args.width
    cap_height = args.height

    use_static_image_mode = args.use_static_image_mode
    min_detection_confidence = args.min_detection_confidence
    min_tracking_confidence = args.min_tracking_confidence

    use_brect = True

    # Camera preparation ###############################################################
    cap = cv.VideoCapture(cap_device)
    cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
    cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

    # Model load #############################################################
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(
        static_image_mode=use_static_image_mode,
        max_num_hands=5,
        min_detection_confidence=min_detection_confidence,
        min_tracking_confidence=min_tracking_confidence,
    )

    keypoint_classifier = KeyPointClassifier()

    point_history_classifier = PointHistoryClassifier()

    # Read labels ###########################################################
    with open('model/keypoint_classifier/keypoint_classifier_label.csv',
              encoding='utf-8-sig') as f:
        keypoint_classifier_labels = csv.reader(f)
        keypoint_classifier_labels = [
            row[0] for row in keypoint_classifier_labels
        ]
    with open(
            'model/point_history_classifier/point_history_classifier_label.csv',
            encoding='utf-8-sig') as f:
        point_history_classifier_labels = csv.reader(f)
        point_history_classifier_labels = [
            row[0] for row in point_history_classifier_labels
        ]

    # FPS Measurement
  ########################################################
    cvFpsCalc = CvFpsCalc(buffer_len=10)

    # Coordinate history
  #################################################################
    history_length = 16
    point_history = deque(maxlen=history_length)

    # Finger gesture history
  ################################################
    finger_gesture_history = deque(maxlen=history_length)

  # Key Points
    for index, landmark in enumerate(landmark_point):
        if index == 0:  # wrist 1
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 1:  # wrist 2
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 2:  # Thumb: base
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 3:  # Thumb: 1st joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 4:  # Thumb: Fingertip
            cv.circle(image, (landmark[0], landmark[1]), 8, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0)
  , 1)
        if index == 5:  # Index Finger: Base
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 6:  # Index finger: 2nd joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
  255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
  , 1)
        if index == 7:  # Index finger: 1st joint
```

```
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 8:  # Index finger:fingertip
            cv.circle(image, (landmark[0], landmark[1]), 8, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0)
    , 1)
        if index == 9:  # Middle finger: base
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 10:  # Middle finger: 2nd joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 11:  # Middle finger: 1st joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 12:  # Middle finger: fingertip
            cv.circle(image, (landmark[0], landmark[1]), 8, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0)
    , 1)
        if index == 13:  # Ring finger: base
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 14:  # Ring finger: 2nd joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 15:  # Ring finger: 1st joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255,
    255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0)
    , 1)
        if index == 16:  # Ring finger: fingertip
            cv.circle(image, (landmark[0], landmark[1]), 8, (255,
    255, 255),
                      -1)
```

```python
            cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)
        if index == 17:  # Little finger: base
            cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
        if index == 18:  # Little finger: 2nd joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
        if index == 19:  # Little finger: 1st joint
            cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
        if index == 20:  # Little finger: fingertip
            cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
                      -1)
            cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)

    return image


def draw_bounding_rect(use_brect, image, brect):
    if use_brect:
        # Outer rectangle
        cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[3]),
                     (0, 0, 0), 1)

    return image


def draw_info_text(image, brect, handedness, hand_sign_text,
                   finger_gesture_text):
    cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[1] - 22),
                 (0, 0, 0), -1)

    info_text = handedness.classification[0].label[0:]
    if hand_sign_text != "":
        info_text = info_text + ':' + hand_sign_text
    cv.putText(image, info_text, (brect[0] + 5, brect[1] - 4),
               cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv.LINE_AA)

#    if finger_gesture_text != "":
#        cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
#                   cv.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 4, cv.LINE_AA)
#         cv.putText(image, "Finger Gesture:" + finger_gesture_text,
    (10, 60),
#                    cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
#                    cv.LINE_AA)

    return image


def draw_point_history(image, point_history):
    for index, point in enumerate(point_history):
        if point[0] != 0 and point[1] != 0:
            cv.circle(image, (point[0], point[1]), 1 + int(index / 2)
    ,
                      (152, 251, 152), 2)

    return image


def draw_info(image, fps, mode, number):
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.
   FONT_HERSHEY_SIMPLEX,
               1.0, (0, 0, 0), 4, cv.LINE_AA)
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.
   FONT_HERSHEY_SIMPLEX,
               1.0, (255, 255, 255), 2, cv.LINE_AA)

    mode_string = ['Logging Key Point', 'Logging Point History']
    if 1 <= mode <= 2:
        cv.putText(image, "MODE:" + mode_string[mode - 1], (10, 90),
                   cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1,
                   cv.LINE_AA)
        if 0 <= number <= 9:
            cv.putText(image, "NUM:" + str(number), (10, 110),
                       cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255),
    1,
                       cv.LINE_AA)
    return image


if __name__ == '__main__':
    main()
```

Appendix B

Screen shots

B.1 Output

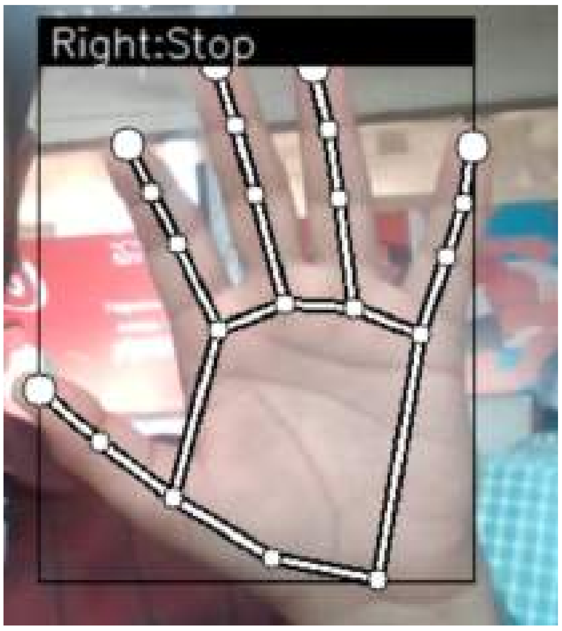

Figure B.1: Stop

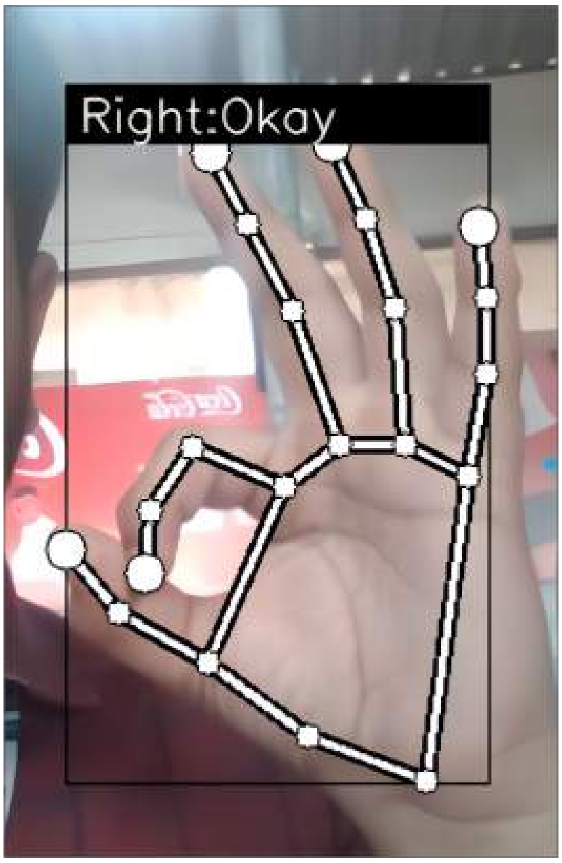

Figure B.2: Enter Caption

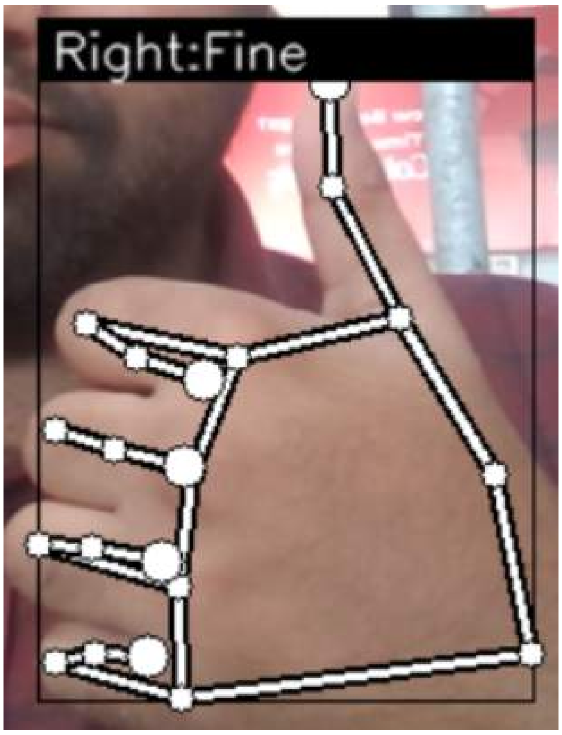

Figure B.3: Enter Caption

Appendix C

Data sets used in the project

The project utilized the IISL2020 dataset, which was created under natural conditions without additional aids like gloves or sensors and consisted of 11 words with approximately 1100 video samples per word from 16 subjects aged 20–25 [3]. The dataset has an average of 28 FPS, a resolution of 1920 x 1080, and an average video length of 2 seconds. Experimental results showed that the model outperformed other sign language datasets, achieving high precision, recall, and F1-scores [4]. The proposed methodology, a combination of LSTM followed by GRU, achieved 97% accuracy in sign identification from the ISL custom dataset (IISL2020) [5] [6]. The dataset can be accessed with the authors' permission [3]. The project also referenced other datasets, like AUTSL and GSL, for benchmarking and comparison [7]. Future research suggestions include improving dataset classes, word sets, and video samples, as well as exploring continuous sign language recognition and addressing challenges like varying brightness levels [8]. The model achieves an impressive accuracy rate of around 97% on 11 different signs in Indian Sign Language, showcasing its potential to bridge communication barriers for individuals with speech or hearing impairments. The study also delves into related work within the field of sign language recognition, highlighting the importance of ongoing research and advancements in this area.The methodology employed in the study involves utilizing a combination of LSTM and GRU models for Indian sign language recognition without the need for specific environments or camera setups. We segment the video files into distinct sections, extract features using InceptionResNetV2, and then pass them through LSTM and GRU layers for processing.

We utilized the IISL2020 dataset, which consists of 11 words and 1100 video samples generated by 16 subjects under natural conditions. The model demonstrates optimized performance in recognizing isolated Indian signs in real-time scenarios, showcasing its practical applicability.