PROJECT REPORT

on

Real-Time Sign Language to Speech Conversion on Edge Devices: A Jetson Nano Implementation

Submitted in partial fulfillment of the requirements for the award of the degree
Of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Submitted By:

Abantika Dasgupta - 00801012021
Bhumika Gupta - 03401012021
Manya Chandna - 03601012021

Under the Supervision(s) of

Dr. Ravinder M.
Department of CSE, IGDTUW
&
Prof. SRN Reddy
HOD, Department of CSE, IGDTUW

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN
KASHMERE GATE, DELHI-110006

April 2025

ACKNOWLEDGEMENT

All of us gratefully acknowledge the valuable and beneficial guidance received from our project guide, ***Dr. Ravinder M. & Prof. SRN Reddy***, Department of Computer Science and Engineering, Indira Gandhi Delhi Technical University for Women, Delhi.

From time to time in the course of preparation of this project, we could bring our efforts to this shape only through the active and keen interest taken by him throughout. We would also like to thank others for their direct or indirect contribution in the successful completion of this project.

Name: Manya Chandna
Enrollment No: 03601012021

Name: Abantika Dasgupta
Enrollment No: 00801012021

Name: Bhumika Gupta
Enrollment No: 03401012021

UNDERTAKING REGARDING ANTI-PLAGIARISM

We, ***Manya Chandna, Abantika Dasgupta & Bhumika Gupta,*** hereby, declare that the material/ content presented in the project report is free from plagiarism and is properly cited and written in our own words. We are fully aware about the UGC regulations for promotion of academic integrity and prevention of plagiarism in higher educational institutions. In case, plagiarism is detected at any stage, we shall be solely responsible for it. A copy of the Plagiarism Report is also enclosed.

Name: Manya Chandna
Enrollment No: 03601012021

Name: Abantika Dasgupta
Enrollment No: 00801012021

Name: Bhumika Gupta
Enrollment No: 03401012021

B. Tech, CSE, 8th Sem

CERTIFICATE

This is to certify that the project entitled ***Real-Time Sign Language to Speech Conversion on Edge Devices: A Jetson Nano Implementation*** is an authentic record of our own work, under the supervision of **Dr. Ravinder M. & Prof. SRN Reddy,** Department of CSE, Indira Gandhi Delhi Technical University for Women, Delhi. It is submitted in partial fulfillment of the requirements for the award of the Bachelor of Technology degree in Computer Science and Engineering at Indira Gandhi Delhi Technical University for Women, Delhi, India. We further certify that:

- This work has not been submitted to any other Institution for any sort of degree/diploma/certificate in India or abroad.
- We totally abided by the plagiarism guidelines given to us by the university for writing and preparing this report.
- Whenever we have used materials (text, data, theoretical analysis/equations, codes/program, figures, tables, pictures, etc.) from other sources, we have done proper citation, linking of the owner's rightful source and giving due credit wherever needed.

Name: Manya Chandna
Enrollment No: 03601012021

Name: Abantika Dasgupta
Enrollment No: 00801012021

Name: Bhumika Gupta
Enrollment No: 03401012021

This is to certify that the project has been done under my supervision and guidance. It has not been submitted elsewhere either partial or full, for award of any degree or diploma to the simplest of my knowledge and belief.

Dr. Ravinder M.
Professor, CSE

Prof. SRN Reddy
HOD, CSE

ABSTRACT

Seamless communication is paramount in empowering the deaf and hard-of-hearing individuals, with sign language acting as a vital conduit for expression. To enhance their societal interactions with non-signers, this study pioneers the development of an advanced application using Tkinter for recognizing words in American Sign Language (ASL) through fingerspelling and subsequently forming sentences leveraging the computational prowess of Jetson Nano. The resulting sign-to-text output is then converted into audible speech through a USB sound adapter connected to the Jetson Nano. To enhance user experience and inclusivity, the system supports text-to-speech conversion in three languages—English, Hindi, and Tamil—based on the user's language preference. The core of the system uses a transfer learning approach using MobileNetV2 as the base model and fine-tuned with custom layers on top of it. The model is trained on a self-curated dataset captured in real-time through a webcam, ensuring robustness and adaptability to practical use cases. The model achieved a training accuracy of 99.8% and a testing accuracy of 93%, highlighting its efficiency in recognizing diverse hand gestures under varying lighting conditions and signing styles. By integrating gesture recognition and multilingual speech synthesis, the system creates a complete communication loop between signers and non-signers. The successful implementation of the application on the NVIDIA Jetson Nano highlights its capability for efficient real-time performance, lightweight deployment, and user-friendly interface. This work marks a meaningful advancement toward inclusive, technology-driven communication solutions, effectively bridging the gap between the deaf and hard-of-hearing community and the hearing world.

Keywords: NVIDIA Jetson Nano, Sign Language Recognition, Deep Learning

LIST OF TABLES

Table No.	Contents	Page No.
2.1.	Overview of Sign Language Recognition approaches and performance	7-10
2.2	Comparison of various computational edge devices	12-13
6.1	Comparison of Model Performance	32
8.1	Key Test Cases	42-43
9.1	Comparison between previous and proposed work	49

LIST OF FIGURES

Figure No.	Figure Title	Page No.
1.1	American Sign Language	2
2.1	Architecture of Jetson Nano	11
4.1	Hand Landmarks detection using Mediapipe	17
5.1	Sign Language Recognition Workflow	20
5.2	Use Case Diagram	22
5.3	DFD Level 0 diagram	22
5.4	DFD level 1 diagram	23
5.5	Sequence diagram	24
6.1	Capturing Raw Image	28
6.2	Hand Landmark Points	29
6.3	Steps of Data Processing	30
6.4	MobileNetV2 Architecture	33
6.5	Model Architecture and associated parameters	34
6.6	Application GUI	37
7.1	Application Deployment Steps	38
7.2	NVIDIA Jetson Nano Setup with USB Speakers attached	39
9.1	Model Accuracy	46
9.2	Model Loss	47
9.3	Confusion Matrix for Classification	48

TABLE OF CONTENTS

Topic	Page No.
Acknowledgment	ii
Undertaking under Anti-Plagiarism	iii
Certificate	iv
Abstract	v
List of Tables	vi
List of Figures	vii

S.No.	Contents	Page No.
1.	**Introduction**	**1**
	1.1. The Importance of American Sign Language	1
	1.2. Challenges in ASL Fingerspelling Recognition	2
	1.3. Advancing ASL Recognition Through Deep Learning	3
	1.4. Research Contributions and Impact	4
2.	**Literature Survey**	**5**
	2.1. Overview of Recent Advances and Challenges in Sign Language Recognition	5
	2.2. SLR with NVIDIA Jetson Nano: A Powerful and Compact AI Solution for Real-Time Gesture Analysis	11
3.	**Problem Identification**	**14**
	3.1 Limited Real-Time ASL Recognition Systems on Edge Device	14
	3.2 Integration of Sign Language Recognition with Text-to-Speech	14
	3.3 Sentence-Level Understanding in Sign Language Recognition	15
	3.4 Data Scarcity and Lack of Diverse Datasets	15
4.	**Proposed Solution**	**16**

	4.1. Dataset Development	16
	4.2. Model Training	17
	4.3 Real-Time Gesture Recognition and Text Generation	18
	4.4 Text-to-Speech (TTS) Module	18
	4.5 GUI Development	19
5.	**Design of Proposed Solution**	**20**
	5.1. Design Objectives and System Goals	20
	5.2 System Design	21
	5.2.1 Use Case Diagrams	21
	5.2.2 Sequence Diagram	24
	5.3 System Architecture Overview	25
	5.4 Performance Optimization	26
6.	**Implementation**	**27**
	6.1. Data Collection	27
	6.2. Data Preprocessing	29
	6.3. Model Training	31
	6.3.1. Model Selection (InceptionV3 to MobileNetV2)	31
	6.3.2. MobileNetV2 Architecture	32
	6.3.3. Customized Layers	33
	6.3.4. Optimization Process	35
	6.4. Real-Time Prediction Pipeline	35
	6.4.1. Steps in Real-Time Prediction	35
	6.4.2. Performance Optimization	36
	6.5. Text-to-speech and GUI Development	36
7.	**Application Deployment on NVIDIA Jetson Nano**	**38**
	7.1. Environment Setup	38
	7.2. Input Handling and Frame Processing	39
	7.3. Sign Recognition and Output Generation	40
	7.4. System Execution and Testing	40
	7.5. Deployment Issues and Challenges	41
	7.6. Feedback from Users	42

8.	**System Testing and Performance Evaluation**	**43**
	8.1 Testing Objectives	43
	8.2 Test Cases and Results	43
	8.3. Testing Metrics	44
	8.4 Observations from Testing	45
	8.5. Improvements Post-Testing	45
	8.6. Testing Summary	46
9.	**Results and Discussions**	**47**
10.	**Conclusion and Future Work**	**51**
	References	**53-56**
	Appendix	**57-60**
	Publication Details	**60-65**
	Plagiarism Report	**66**

CHAPTER 1
INTRODUCTION

Overview: *This chapter introduces the motivation behind developing a real-time ASL recognition system, emphasizing the communication gap between signers and non-signers. It outlines the importance of ASL, the challenges of fingerspelling recognition, and how deep learning techniques—particularly MobileNetV2—are used to address these issues. The chapter also highlights the key contributions of the study, including dataset creation, model training, and deployment on edge hardware for real-time performance.*

Sign language is an essential medium of communication for the hard-of-hearing community, allowing individuals to express complex ideas and emotions through hand gestures, facial expressions, and body movements.. In contrast to spoken languages that rely on auditory impulses, sign language is visual and dimensional, providing a rich and expressive way of engaging individuals. However, despite its increasing significance, sign language is not widely understood by the general masses. This leads to communication barriers between the sign language users and the non signers that can result in frustration, social alienation and limited outlook for the hearing impaired community. By bridging this communication barrier through technological improvements, can in turn promote inclusivity and improve accessibility.

1.1 The Importance of American Sign Language (ASL)

American Sign Language (ASL) is a fully developed, natural language utilized primarily by the Deaf community in the United States and parts of Canada. This is a complete linguistic system as a whole, with its own grammatical structure, as opposed to English and other spoken languages. ASL is based on a wide range of hand signs, movements, and facial expressions to convey an extensive spectrum of information. The ASL signs for English alphabets are illustrated in Fig 1.1.

One of the critical aspects of ASL is fingerspelling, involving the use of different hand shapes to represent individual alphabets of the English alphabet. It is in turn essential for spelling out proper names, technical terminology, and words that do not have a designated sign for it to be conveyed. By automating the recognition of ASL through the fingerspelling method and its progressive conversion into text and then speech, we can significantly enhance accessibility and facilitate convenient communication between signers and non-signers. This study proposes a real-time ASL recognition system utilising computer vision and deep learning techniques to interpret ASL fingerspelling and converting it into readable formats in real time.

Fig. 1.1 American Sign Language

1.2 Challenges in ASL Fingerspelling Recognition

Real- time recognition of ASL through fingerspelling defines several technical and practicality challenges with some of the major difficulties inclusive of:

- **Variability in Hand Gestures**: Subtle differences in the positioning of the finger, its orientation, and subsequent movement can lead to ambiguity between similar letters, like “M” and “N.”

- **Environmental Effects**: Changes in lighting conditions, background noise, camera angles, as well as hand placement can affect recognition accuracy when operating in varying settings.
- **Real-Time Processing Necessity**: Effective communication tools have to process sign language rapidly for maintaining natural interaction, further requiring optimized models and increased computational efficiency.
- **Diversity in Hand Features**: Different hand sizes, skin tones, and personal signature styles can impact model output and performance thus requiring extensive dataset diversity for reliable recognition.

Addressing these challenges requires a diverse dataset as well as robust machine learning techniques capable of differentiating subtle visual patterns while maintaining high accuracy and speed at the same time. Additionally, the inclusion of computational edge AI devices such as the NVIDIA Jetson Nano allows for real-time processing, making it effectively reliable to deploy this system for practical applications and use cases.

1.3 Advancing ASL Recognition Through Deep Learning

Over several years, traditional methods of sign language recognition have included rule-based and distinctive feature extraction methods. However, recent advances in deep learning, particularly in the field of Neural networks, particularly, Conventional Neural Networks (CNNs), have significantly improved upon the various visual recognition capabilities.

This research focuses on using MobileNetV2 as the base model, an efficient lightweight CNN architecture, for ASL fingerspelling recognition. By fine-tuning the model with additional custom layers, we fulfill the objective of classifying ASL letters accurately. A comprehensive dataset of ASL hand gestures was formulated to train the model, recording various lighting conditions, hand positions, and backgrounds to ensure robustness and efficiency. The system processes real-time input from an integrated webcam, further recognizes letters, and gradually forms words and sentences with an additional speech output to promote seamless interaction. The final implementation is optimized for edge AI deployment, ensuring low-latency performance on the Jetson Nano hardware platform.

1.4 Research Contributions and Impact

This study offers several contributions to the advancement of ASL recognition practice:

- **Development of a Custom ASL Dataset**: A specialized vast dataset of American Sign Language hand gestures was created to enhance recognition accuracy and improvise model training.
- **Implementation of an Optimized Deep Learning Model**: The MobileNetV2 architecture was used as a base model and was further modified with custom layers to enhance classification performance.
- **Real-Time Recognition System**: A functional ASL-to-Speech translation system was designed, enabling seamless communication between signers and non-signers.
- **Multilingual Support:** The final speech output can be given out any of the three languages integrated, that is, English, Hindi and Tamil.
- **Deployment on Edge AI Hardware**: The system was deployed on NVIDIA Jetson Nano, allowing for efficient, low-latency processing suitable for applications in real-time environments.
- **User-Friendly GUI Integration**: A graphical interface was developed using Tkinter, allowing users to interact with the system effectively and view recognized text output instantly.

By addressing the communication barriers faced by the hard-of-hearing community, this research promotes easier accessibility and social inclusivity. The proposed research lays the groundwork for further future developments in the field of sign language recognition, with potential applications in assistive communication devices and applications, ASL awareness and education, and broader accessibility technological solutions. This study underscores the transformative role of the domain of AI and computer vision in fostering inclusivity and bridging the communication gap between signers and non-signers.

CHAPTER 2
LITERATURE SURVEY

Sign Language Recognition has become a pivotal area of research, aiming to bridge the communication divide among the Deaf community that uses sign language for communication and individuals unable to use sign language. Taking into account the increasing development in deep learning and computer vision, attempts have been driven towards developing highly effective and accurate systems that function in real-time while simultaneously addressing the inherent complexities of sign language, such as its unique grammar, temporal variability, and environmental impacts. The field has evolved from isolated sign recognition to more sophisticated and enhanced continuous and word-level recognition models, expanding the possibilities to a wider spectrum for improved accuracy and practical implementation.

2.1 Overview of Recent Advances and Challenges in Sign Language Recognition

SLR has made remarkable advancements in recent times, devising various methods to equip the systems with better accuracy. For instance, [1] developed a deep convolutional network for ASL alphabet and digits with an accuracy of 82.5%. [2] analyzed video-based human representations for ASL letter recognition utilizing transformers, emphasizing the importance for large datasets and intercepting computational challenges. [3] introduced a transformer-based model aiming at recognizing ASL through fingerspelling, essentially addressing alphabet recognition in natural environment, with difficulties for overlapping gestures. Similarly, [4] developed a lightweight MobileNetV2 CNN-based architecture for recognizing ASL and Indian Sign Language (ISL) letter, with a commendable 98.77% accuracy.

Beyond mere alphabet recognition, significant research has been conducted on word-level SLR such as, [5] reviewed several deep learning methodologies for SLR, emphasizing on the need of high accuracy for isolated signs, yet discovering ongoing challenges for continuous recognition due to complexities in grammar structure of sign language. While [6] examined temporal

modeling architectures, such as 3D CNNs and temporal convolutional networks, [7] presented a multi-feature network integrating hand shapes, movements, and body postures, achieving strong performance on the infamous PHOENIX-2014T dataset.

In continuous SLR, [8] introduced a transformer-based model that learns both recognition and its translation simultaneously. Meanwhile [9] created a framework to formulate realistic sign language videos from text. The exploration of RGB-D sensors for continuous SLR was introduced in [10], with a combined hand shape and body pose information, yielding impressive results on Turkish Sign Language datasets. Recent developments also feature transformer-based architectures using spatial and temporal data in [11], as well as a zero-shot learning approach to recognize signs that have not been previously identified [12]. Integration of linguistic elements into SLR models [13] further improves recognition efficiency.

[14] highlights the necessity of user centric design in SLR technologies, emphasizing the urgency to involve Deaf individuals in the development procedure to create practical and accessible solution. Additionally, attention mechanisms have enhanced model accuracy [15]. Challenges such as data scarcity and domain adaptation have also been tackled. [16] introduced a multi-channel transformer to manage variations in signing styles and camera angles, while [17] concentrated on cross-dataset domain adaptation to enhance real world performance. Large-scale datasets, such as MS-ASL [18], have substantially provided training data for the models.

The fusion of Motion trajectory data with CNN and LSTM networks further refined dynamic sign capture [19]. [20] devised a real-time ASL recognition system utilizing the capabilities of Mediapipe and CNNs, achieving high accuracy for word-level SLR, though restrained to predefined gestures. Lastly, [21] focuses on combining Convolutional Neural Networks (CNN) and other deep learning techniques for accurate recognition, while [22] further addressed the challenge of real-world application leveraging CNNs. We have reviewed various deep learning methodologies for SLR, a summary of which is given in **Table 2.1.**

Table 2.1 Overview of Sign Language Recognition approaches and performance

Ref	Type of Sign Language	Algorithm Used	Model Used	Dataset	Testing Accuracy (%)	Limitations/Grey areas
[1]	ASL (Alphabets)	Deep Convolutional Networks	CNN	ASL Alphabet dataset	82.5%	1. Does not emphasize the development of real-time application and primarily focuses on the training and validation phases. 2. The model may be overfitting to specific training conditions and needs to be generalized well to diverse environments.
[2]	ASL (Alphabets)	Video based Human Representations	Transformer	RWTH-PHOENIX-Weather 2014T	87.5%	High computational cost, requires large-scale datasets
[3]	ASl (Alphabets)	Object Detection Network	Transformer	Chicago Fingerspelling in the Wild	92.6%	Struggles with overlapping signs and complex hand movements
[4]	ASL, ISL (Alphabets)	Deep Learning Based	Light weight CNN; Mobil	Akash Asl alphabet : 87,000	98.77%	1. Doesn't consider factors such as varying lighting, background noise, or occlusions.

			eNet V2	images & ISL-CSLTR		2. Confusion between similar hand signs, such as the letters "P" and "Q,"
[7]	Continuous Sign Language (Words)	Spatial-Temporal Multi-Cue Network	CNN-LSTM	RWTH-PHOENIX-Weather: 75,000 frames	85.4%	1. The complexity of handling multiple cues (hand shape, facial expressions, body posture) presents challenges 2. Relies on video-based inputs for CSLR, which poses limitations in environments with variable lighting, occlusions, or inconsistent background conditions.
[8]	Continuous Sign Language (Words)	Sign Language Transformer	Transformer	RWTH-PHOENIX-Weather: 825,000 frames	85.4%	1.RWTH-PHOENIX-Weather-2014T dataset is specifically focused on weather data, which limits the diversity of signs 2. Employs a transformer-based architecture which is computationally intensive.
[10]	Turkish Sign Language (Words)	CNN-LSTM	3D CNN + LSTM	AUTSL Dataset: 226 sign classes	91.47%	1. Inclusion of Kinect-based recordings limits its generalizability to other real-world settings

						2. Highlights a limitation in the model's ability to capture fine-grained distinctions between similar gestures
[11]	Word-level Sign Language (Words)	CNN-based methods	ResNet-50	WLASL dataset: 21,000 signs	93.0%	1. Decrease in performance as the vocabulary size increases. 2. reliance on visual cues(appearance-based approaches) can be problematic in real-world settings, where lighting and background conditions may vary widely
[18]	American Sign Language (Words)	Deep learning-based	MS-ASL Benchmark	MS-ASL dataset: 1,000 classes	94.5%	1. Models perform well in recognizing individual signs, but they still struggle with contextual understanding. 2. The dataset includes videos in unconstrained environments, with significant challenges such as variability in background, lighting, and camera angles

[19]	Indian Sign Language (Gestures/ Continuous Words)	3D CNN	ResNet	ISL Gesture dataset: 10,000 gestures	87%	1. Sensor Limitation reliance on specific 3D hand gesture sensors like Kinect, Leap Motion, and Time of Flight (ToF) 2. System struggles with inter-hand variations and occlusions, leading to non-discriminatory features.
[20]	ASL (Words)	Mediapipe and CNNs	CNN-based	Custom dataset for ASL gestures	95.4%	Limited to predefined gestures, scalability issues
[21]	ASL(Alphabets)	Hybrid	CNN-based	ASL Alphabet dataset	71.4%	1.Struggles to differentiate between certain similar handshapes 2.Doesn't explicitly account for the fluency and transitions between handshapes in fingerspelling
[22]	ASL (Alphabets)	CNN	CNN-based	Custom dataset for ASL gestures	98.84%	Dependency on Predefined Image Resolutions

2.2 Sign Language Recognition with NVIDIA Jetson Nano: A Compact yet powerful AI Solution for Real-Time Gesture Analysis

The NVIDIA Jetson Nano stands out as a compact and commanding AI technology, designed specifically for applications in Sign Language Recognition (SLR). Its capability to utilize multiple neural networks at the same time makes it an effective solution for processing gesture-based inputs, allowing for real-time sign language gesture recognition into text and speech. It is designed with impressive adaptability and scalability in mind, allowing seamless integration into compact systems for gesture recognition. It manages high-resolution sensor inputs and performs parallel deep-learning processing, making it ideal for SLR procedures. Supported by the NVIDIA JetPack SDK and a wide range of inbuilt libraries, the Jetson Nano is well-equipped to handle Gesture Recognition and Tracking. Its 128-core Maxwell GPU utilizing CUDA, delivers exceptional computational efficiency for processing visual data from integrated cameras. Furthermore, ARM Cortex-A57 quad-core CPU works alongside GPU, creating a user friendly hybrid environment for AI-driven applications.

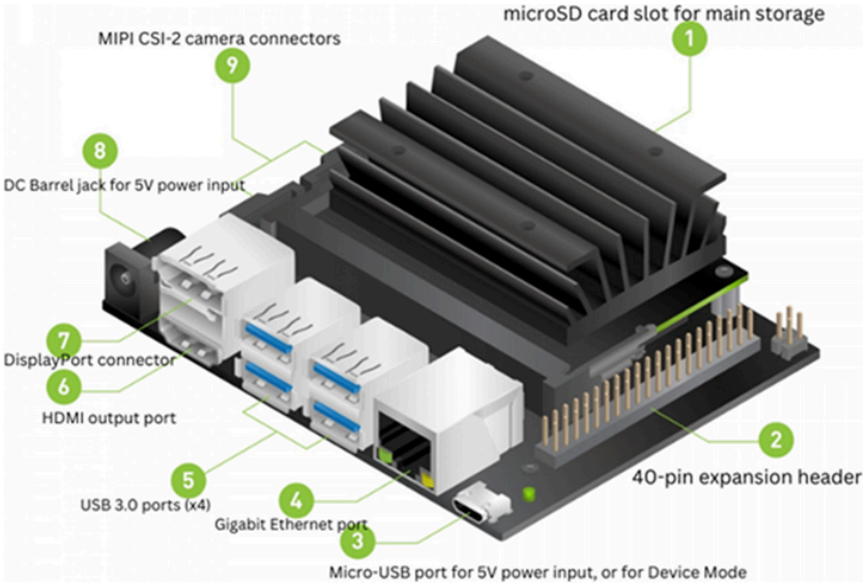

Fig. 2.1 Architecture of Jetson Nano

Fig. **2.1** presents the architecture of Jetsan Nano, a computing device perfectly tailored for edge computing. The NVIDIA Jetson Nano Developer Kit comes equipped with a compact yet

versatile set of I/O interfaces, making it ideal for embedded AI applications like Sign Language Recognition. These physical components allow developers to connect cameras, display modules, sensors, and other peripherals seamlessly, enabling full deployment of AI-based gesture recognition systems directly at the edge.

Jetson Nano Ideality for SLR Systems: After thoroughly evaluating various computational modules, as displayed in table 2.2, we have conclusively chosen the Jetson Nano. This choice is based on its optimal equilibrium between the need for fast processing and constraints of computational resources and memory. With a Quad-Core ARM®-based CPU and the powerful NVIDIA Maxwell™ architecture GPU, the Jetson Nano functions as a portable server with remarkable capabilities. System proposed in [23] showcased the Jetson Nano's effectiveness in real-time SLR applications, underscoring its appropriateness for such tasks. The main reason lies in its ability to balance cost- effectiveness and strong CPU performance. We need a system that is both budget friendly and capable of meeting performance requirements, while also being easily accessible when necessary. Designed to integrate seamlessly into the Internet of Things (IoT) ecosystem, it is about the size of a mobile phone, providing a variety of deployment alternatives in connected environments. The affordable price, high performance for neural networks and portable nature of Jetson Nano make it an excellent choice for SLR applications. Additionally, it supports efficient edge computing, which helps the sign language models to function in real-time without relying on cloud systems.

Table 2.2 Comparison of various computational edge devices

Equipment	**NVIDIA Jetson Nano [24]**	**Raspberry Pi 4[25]**	**Google Coral Dev Board[26]**	**Jetson Xavier NX[27]**	**Jetson AGX Xavier[28]**
CPU	ARM A57	ARM Cortex-A72 64-bit	NXP i.MX 8M SOC	8-core ARM	integrated ARM Hexa-Core CPU
No. of cores	4	4	4	8	6

	Freq: 1.43 GHz	Freq: 1.5 GHz	Freq: 1.5 GHz	Freq: 2.26 GHz	Freq: 2.0 GHz
GPU	128-core Maxwell	Broadcom VideoCore VI	Integrated GC7000 Lite graphics	512-Core Volta+NVDLA	512-Core Volta+NVDLA
Power	5W-10W	2.56-7.30W	4TOPS-2W	10/15/30 W	30W
RAM	4 GB	1 GB, 2 GB, 4 GB, 8 GB	1 or 4 GB LPDDR4	8 GB	16 GB LPDDR4
Ports	4x USB 3.0	2 USB 3.0 ports	USB 2.0/3.0 ports	(4x) USB 3.1 (Host)	USB 3.1 Type-C, Client Port USB 3.0
Camera	2x MIPI CSI-2	MIPI CSI port	MIPI CSI-2	16 lanes MIPI CSI-2	4 line MIPI, USB Camera port
connectivity	Gigabit Ethernet, Wireless networking adapter	Gigabit Ethernet, RJ45 (WiFi), M.2 Key M (NVMe)	Ethernet, USB Wi-Fi adapter	Gigabit Ethernet, M.2 Key E (WiFi), M.2 Key M (NVMe)	Gigabit Ethernet, GPS, PCIe gen2
Cost	$99	$55	$85.99	$699	$299
OS	Linux4Tegra	Ubuntu Mate, Snappy Ubuntu Core, etc.	ARM and Linux	Linux r35 codeline	Android and Linux

CHAPTER 3
PROBLEM IDENTIFICATION

3.1 Limited Real-Time ASL Recognition Systems on Edge Devices

While numerous advancements have been made in the domain of sign language recognition, many of these models are heavily dependent on powerful GPU-enabled systems or cloud-based processing, which restricts their deployment in real-time, resource-constrained environments. Edge devices like the NVIDIA Jetson Nano present a viable low-cost and portable platform for real-world deployment; however, the limited computational resources impose significant challenges in running complex deep learning models at acceptable frame rates. Our research specifically addresses this bottleneck by employing lightweight architectures (MobileNetV2) and system-level optimizations that allow for efficient real-time performance on edge hardware without compromising recognition accuracy.

3.2. Integration of Sign Language Recognition with Text-to-Speech

A significant limitation in current ASL recognition systems is that they often end at the stage of textual representation of recognized signs. While this textual output is informative, it lacks the auditory dimension that is critical for interactions in broader social settings involving non-signers. Few studies have successfully integrated sign language recognition with a functional text-to-speech (TTS) system, particularly in a real-time setting. Our work bridges this gap by seamlessly converting the recognized sign language into audible speech using a USB sound adapter connected to the Jetson Nano. Moreover, we provide multilingual TTS support—currently offering Hindi, English, and Tamil—thus catering to a diverse user base and enhancing the inclusivity and utility of the system in multilingual societies.

3.3. Sentence-Level Understanding in Sign Language Recognition

The majority of existing ASL recognition systems are restricted to isolated character or word-level recognition, which severely supresses their practical use case in real-life communication scenarios. Human communication using sign language is inherently sequential

and context-based, requiring systems to go beyond single gestures and understand continuous signing as cohesive phrases sentences. This project tackles this issue by implementing a mechanism that gathers recognized letters into words and then constructs meaningful phrases.

3.4. Data Scarcity and Lack of Diverse Datasets

Publicly available datasets for ASL recognition are restrictive in both quantity and diversity, particularly when it comes to lighting variations, background conditions, and signer demographics which hinders the generalizability of trained models. To tackle this, we have introduced a self-curated dataset with webcam-based acquisition under tweaked environmental conditions, capturing extensive hand postures for each letter of the ASL alphabet. This dataset is fit the requirements of our real-time edge deployment scenario and serves to enhance the model's adaptability and accuracy in uncontrolled scenarios. Furthermore, extensive data augmentation methods are applied to enrich the dataset and reduce overfitting, ensuring better performance across unfamiliar data.

CHAPTER 4
PROPOSED SOLUTION

4.1 Proposed Solution

The proposed Sign Language Recognition (SLR) system is designed to address the challenges associated with recognizing American Sign Language (ASL) fingerspelling. The system aims to deliver high accuracy, real-time performance, and robust adaptability to various environmental conditions. The following objectives outline the core components of the system:

4.1.1 Dataset Development

To enable accurate classification of ASL gestures, a comprehensive dataset was developed using a high-resolution USB camera. The dataset was built with careful consideration of real-world variability, capturing gestures under diverse conditions:

- Variations in hand orientation (e.g., palm up/down, tilted angles)
- Lighting diversity, including natural daylight, artificial indoor light, and low-light conditions
- Multiple background scenarios, such as plain walls and cluttered environments

Each recorded video frame underwent preprocessing through MediaPipe Hands, which detects 21 key landmark points on each hand. These skeletal points represent finger joints and palm orientation, forming the basis for consistent feature extraction across samples. This structured representation allows for:

- Geometric analysis of hand poses
- Input normalization across different hand sizes and positions

To further improve model robustness, several data augmentation techniques were applied:

- Rotation and scaling to simulate natural hand movement
- Brightness and contrast adjustment to handle lighting inconsistency
- Gaussian noise addition to increase tolerance to camera artifacts

This augmented dataset significantly enhances the generalization ability of the trained model. Fig. 4.1 below illustrates the 21 hand landmark points captured using MediaPipe, which form the foundation of gesture recognition.

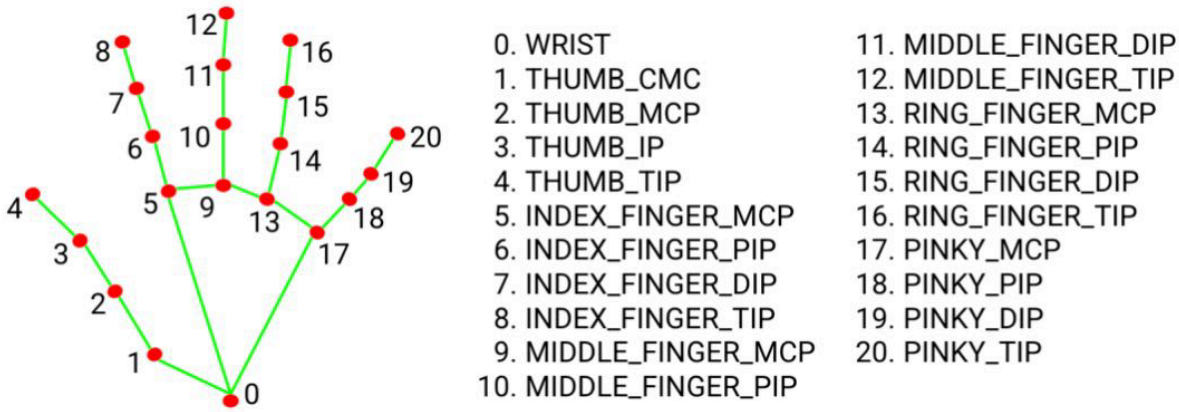

Fig. 4.1: Hand landmarks detection using Mediapipe

4.1.2 Model Training

The SLR model architecture is rooted in deep learning principles, originally employing the **InceptionV3** architecture for high accuracy. However, to better align with the resource constraints of embedded devices such as the Jetson Nano, the model was later optimized and fine-tuned using **MobileNetV2**, a lightweight CNN well-suited for edge deployment.

Key aspects of the model training process include:

- Training on a curated dataset of 25 ASL gestures, excluding the gesture for ‘J’ (dynamic gesture)
- Use of categorical cross-entropy loss function for multi-class classification
- Adam optimizer with a learning rate scheduler for effective convergence
- Batch normalization and dropout layers to prevent overfitting

The model achieved a strong balance between classification accuracy and inference speed, enabling consistent real-time predictions with minimal latency when deployed on NVIDIA Jetson Nano.

4.1.3 Real-Time Gesture Recognition and Text Generation

Once deployed, the system continuously captures live video input and processes each frame to identify and classify hand gestures in real time. The recognition pipeline includes:

- Real-time ROI extraction from video frames
- MediaPipe-based landmark detection
- Gesture classification using the trained MobileNetV2 model

To ensure coherent sentence formation from a series of recognized gestures, a rolling window mechanism is employed. This approach:

- Collects a sequence of recognized characters over a defined frame interval
- Aggregates stable predictions to reduce temporal inconsistencies
- Filters out misclassifications due to transient hand movements

The recognized characters are dynamically converted into words and sentences, with built-in spell-checking using Hunspell and context-based refinement to correct minor errors and enhance output clarity.

4.1.4 Text-to-Speech (TTS) Module

To ensure inclusive communication, the system incorporates a multi-language text-to-speech (TTS) module. Once a sentence is formed from recognized gestures, it is audibly rendered in one of the following supported languages:

- English
- Hindi
- Tamil

The language selection is available through the GUI, and the TTS is powered by pyttsx3, allowing offline speech synthesis without internet dependency. This functionality provides:

- Auditory feedback for the listener

- Enhanced multi-modal communication for the deaf and hard-of-hearing community
- Improved interaction in multilingual environments

4.1.5 GUI Development

To offer a smooth and intuitive user experience, a custom-designed GUI was developed using Tkinter. The interface provides real-time visual feedback by displaying:

- Live camera feed with hand landmark overlays
- Recognized character and word stream updated dynamically
- Full sentence formation and suggested corrections
- An integrated language selector for TTS output

Additional interactive features include:

- Reset controls to clear previous inputs
- Theme customization for visual accessibility
- Session logging for tracking user performance (optional)

The GUI design emphasizes simplicity and ease of navigation, making the system accessible to users of varying technical backgrounds.

CHAPTER 5
DESIGN OF PROPOSED SOLUTION

Overview: *This section outlines the architectural and functional design of the proposed Sign Language Recognition (SLR) system, emphasizing modular integration, performance efficiency, and usability. The design leverages state-of-the-art tools and models to ensure high recognition accuracy, real-time responsiveness, and user accessibility in practical scenarios.*

5.1. Design Objectives and System Goals

The primary objective of the proposed system is to deliver an **accurate, responsive, and accessible solution** for American Sign Language fingerspelling recognition.

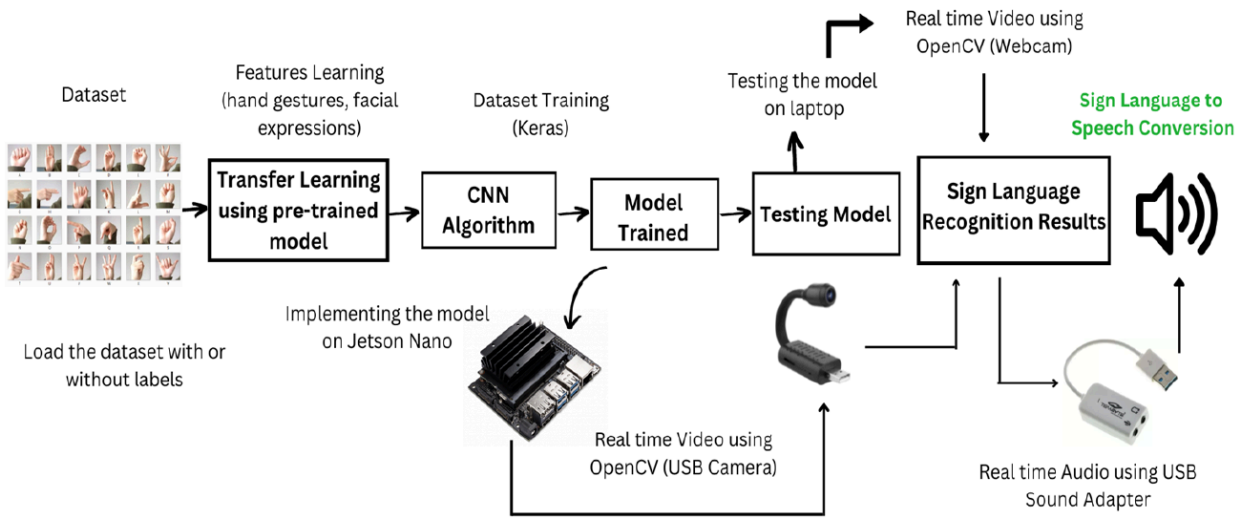

Fig 5.1. Sign Language Recognition Workflow

As illustrated in **Figure 5.1**, the design centers around the following key goals:

- **Minimal Latency**: The system is designed for real-time execution with near-instantaneous gesture recognition, achieved by optimizing the data pipeline and leveraging GPU acceleration on the Jetson Nano.

- **Environmental Robustness**: It maintains consistent performance across varying environmental conditions, such as different lighting levels, cluttered or dynamic backgrounds, and multiple hand orientations.
- **Intuitive User Experience**: Through a simplified and responsive Graphical User Interface (GUI), along with a real-time Text-to-Speech (TTS) module, the system enables a natural and seamless user interaction.
- **Dynamic Sentence Construction**: Using a rolling window approach and post-processing logic, recognized gestures are continuously translated into coherent text and sentences, allowing fluid communication.

These objectives collectively ensure that the system is practical for real-world applications, such as education, communication assistance, and public service accessibility.

5.2 System Design

5.2.1 Use Case Diagrams

The use case diagram, shown in Figure 5.2 below, illustrates the interaction between the primary actors—Signer (ASL User) and Non-Signer (Listener/User)—and the system. The process begins with the signer providing input in the form of a live video feed. The system processes the video frames provided as input by performing character-level prediction using a trained deep learning model. These predicted characters are dynamically combined into words, with a suggestion feature to handle incomplete or uncertain inputs. Once words are formed, the system continues to build complete sentences.

Following sentence formation, the Text-to-Speech (TTS) module generates corresponding audio output. This module supports multilingual speech synthesis, allowing users to select from supported languages such as English, Hindi, or Tamil. Finally, the Non-Signer receives the spoken output, facilitating effective communication between the signer and the non-signer.

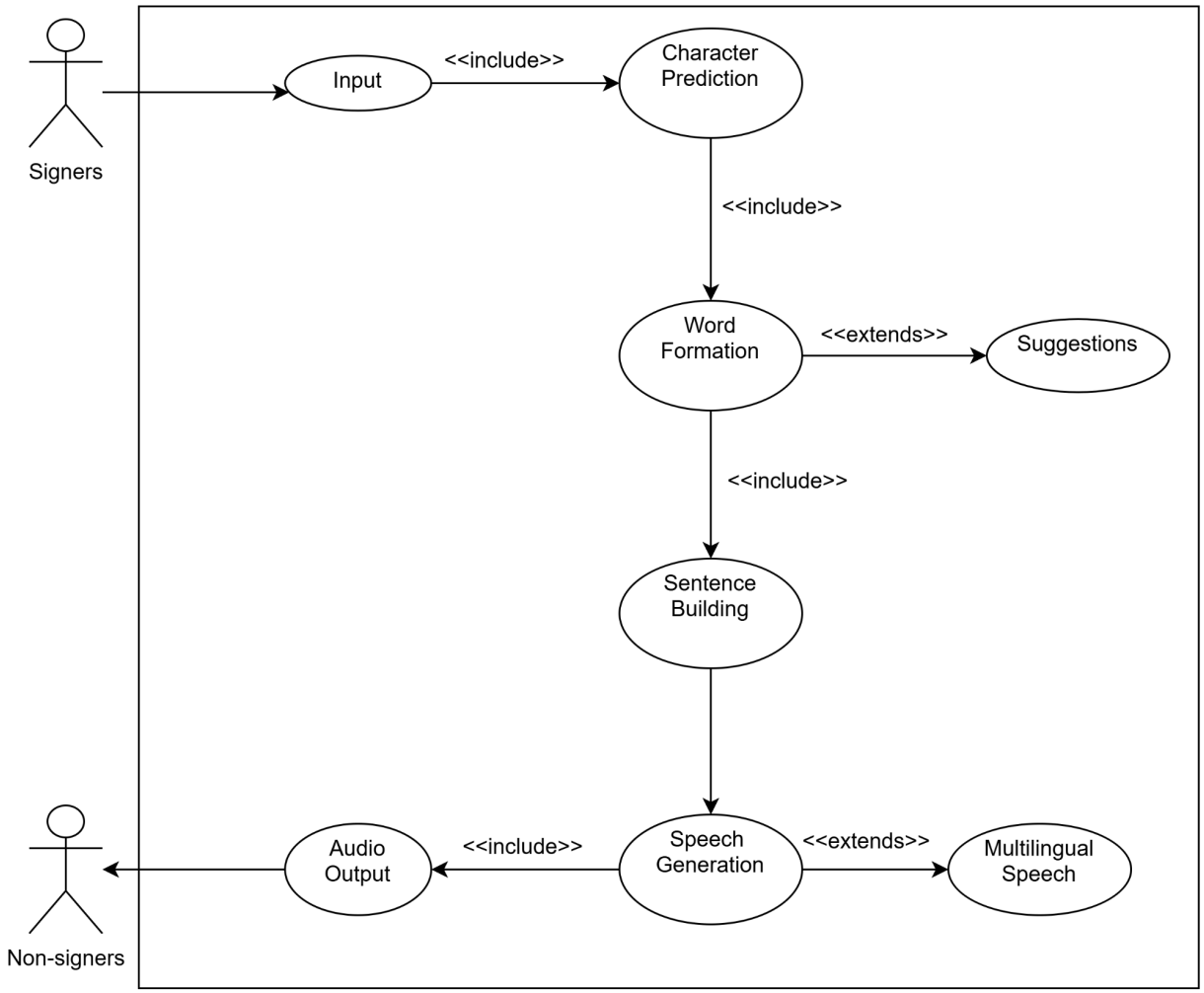

Fig. 5.2: Use Case Diagram

To provide a deeper understanding, Figures 5.3 and 5.4 present the Data Flow Diagrams (DFDs):

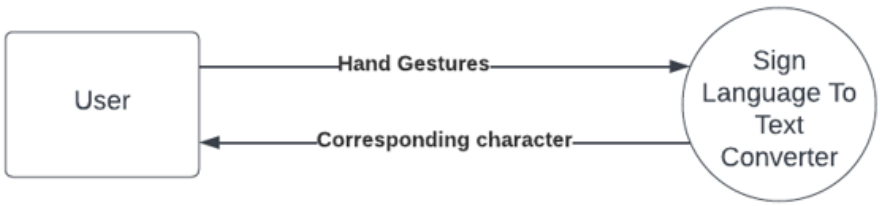

Fig. 5.3: DFD Level 0 diagram

Figure 5.3 (DFD Level 0) offers a high-level overview of the core processes: data capture, gesture classification, and output generation.

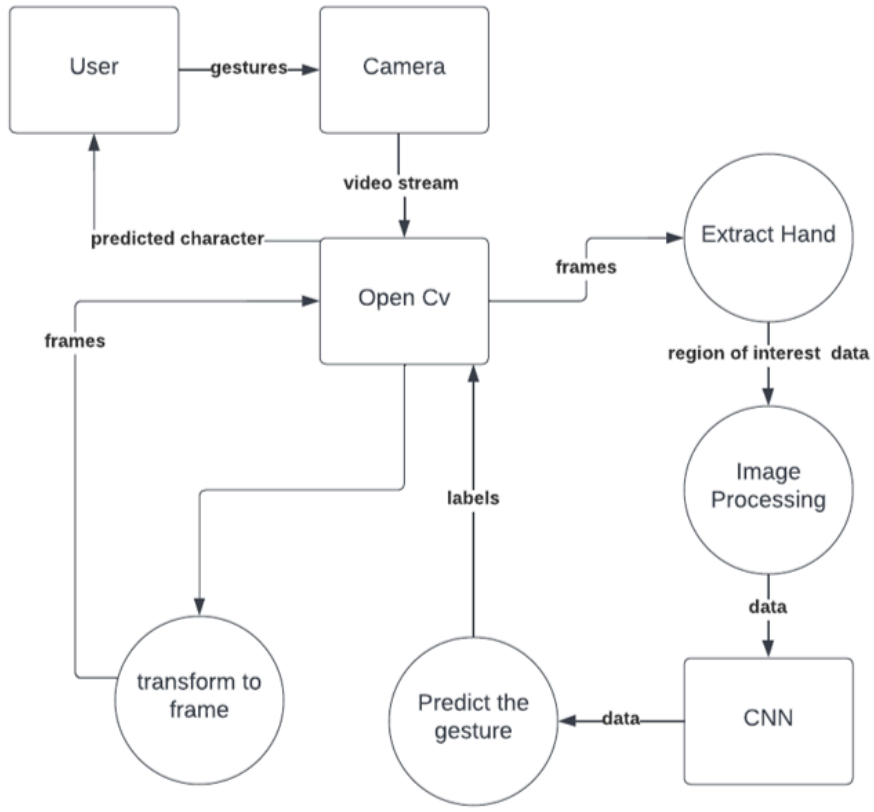

Fig. 5.4: DFD level 1 diagram

Figure 5.4 (DFD Level 1) breaks down these processes further, showing:

- Input acquisition from a USB camera
- Frame extraction and preprocessing
- Landmark detection via MediaPipe
- Gesture classification using MobileNetV2
- Text formation and refinement
- Speech output through the TTS module

5.2.2 Sequence Diagram

The Sequence Diagram in **Figure 5.5** captures the temporal flow of interactions between system modules, illustrating how a user's hand gesture moves through the processing pipeline from detection to output. This includes:

- Gesture input
- MediaPipe detection
- Model prediction
- Text formation and correction
- GUI rendering and speech synthesis

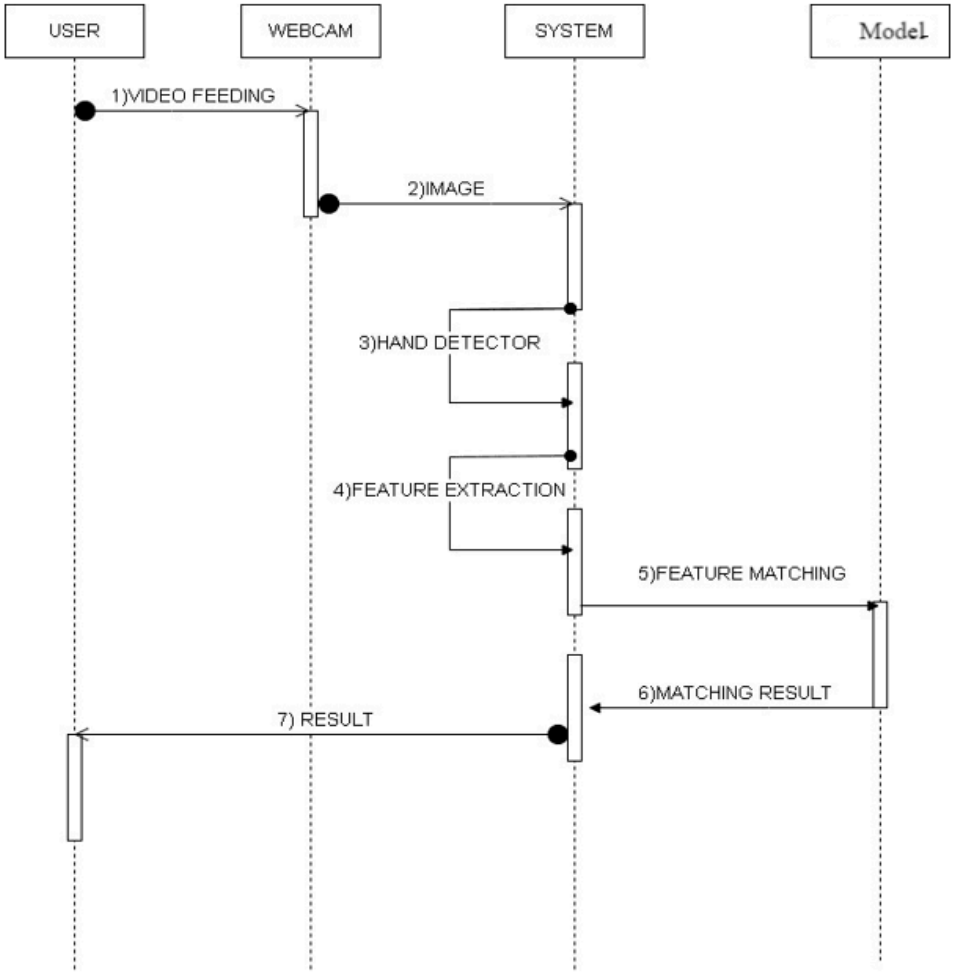

Fig. 5.5 : Sequence diagram

5.3 System Architecture Overview

The architecture of the proposed SLR system is modular and includes five principal components, each responsible for a specific phase of the recognition pipeline:

1. Input Module

- Captures live video stream at 30 frames per second (fps) using OpenCV.
- Interfaces with an external USB camera or the Jetson Nano's CSI camera module.
- Streams video frames into the processing pipeline with minimal latency.

2. Preprocessing Module

- Converts RGB frames to grayscale for computational efficiency.
- Applies Gaussian blur to reduce image noise and binary thresholding for contrast enhancement.
- Extracts 21 hand landmarks using MediaPipe, enabling a structured skeletal representation of the hand.

3. Recognition Module

- Utilizes a fine-tuned MobileNetV2 model trained to classify static ASL gestures (A–Z, excluding J).
- Processes each frame's landmark data to output the most probable character class.
- Optimized for real-time inference with minimal overhead on Jetson Nano.

4. Post-Processing Module

- Implements a rolling prediction window to smooth out transient misclassifications.
- Dynamically assembles characters into words and complete sentences.
- Applies spell-check and context refinement using the Hunspell library to enhance language fluency.

5. User Interface Module

- Displays live video feed with overlaid landmark visualizations.

- Renders predicted characters, words, and sentences in real time.
- Converts the final recognized text into audible speech using a multi-language TTS system (English, Hindi, Tamil).
- Provides options for reset, language selection, and display customization, ensuring accessibility for all user types.

The system workflow follows a sequential logic pipeline starting from video capture and ending with synthesized speech, creating a fluid interaction loop for the end-user.

5.4 Performance Optimization

To ensure both high accuracy and system responsiveness, several optimization strategies were employed:

- **MediaPipe Integration**: MediaPipe's efficient landmark detection reduces the complexity of image-based recognition by focusing on **skeletal hand representations**. By isolating gestures from noisy backgrounds, it enhances recognition precision and minimizes false positives.
- **Background Suppression**: Dynamic background elimination is achieved by masking irrelevant image regions and focusing processing solely on hand features. This increases model focus and reduces inference overhead.
- **Hardware Acceleration**: Leveraging Jetson Nano's 128-core Maxwell GPU and CUDA support, the system accelerates deep learning inference and media processing without overloading the CPU, allowing simultaneous GUI rendering and gesture recognition.
- **Energy Efficiency**: The lightweight model architecture and GPU utilization ensure **low power consumption**, making the system suitable for portable or embedded use cases, such as wearable devices or smart kiosks.

  Enhanced Accessibility: The multi-language TTS and responsive GUI allow the system to cater to users across different linguistic backgrounds and abilities, promoting inclusivity in communication.

CHAPTER 6
IMPLEMENTATION

Overview: *This chapter outlines the structured process followed to implement a real-time American Sign Language (ASL) recognition system, detailing key stages from data collection to system deployment on the Jetson Nano. It explores each critical component and its role in optimizing system performance, ensuring efficient ASL fingerspelling recognition. The chapter begins by discussing the creation of a high-quality ASL dataset, followed by preprocessing steps, model training, and the optimization of real-time prediction pipelines. It also highlights the integration of Text-to-Speech (TTS) functionality, the development of a user-friendly graphical interface (GUI), and the deployment on NVIDIA Jetson Nano for seamless and responsive real-time recognition.*

The system's implementation follows a detailed structured process, from data collection to the system deployment on Jetson Nano, ensuring efficient and real-time ASL fingerspelling recognition. Each stage plays a crucial role in optimizing performance, and this overview indicates the key components and procedures involved in the development.

6.1 Data Collection

The objective is to develop a high-quality dataset representing ASL letters (A–Y, excluding 'J' and 'Z' due to their dynamic nature) with sufficient heterogeneity to support robust model training. An example of capturing a raw image of the alphabet 'W' during data collection is shown in **Figure. 6.1**.

It involved the following steps

1. **Dataset Creation:** A custom dataset of 12,025 training images & 6,525 testing images was collected across 25 distinct classes (24 ASL letters and 1 blank class). A USB camera integrated to Jetson Nano was used to capture mirrored video frames, simulating how end-users typically see themselves. Each letter was captured as a static hand gesture, with dynamic gestures like 'J' and 'Z' excluded due to their dynamic nature.
2. **Environment Setup:** Images were captured under different lighting conditions and backgrounds to simulate real-world environments. Users were instructed to vary hand orientations and positions to regulate dataset diversity. The Jetson Nano's processing

capabilities enabled real-time image capture and subsequent preprocessing, ensuring consistency in the dataset.

Fig. 6.1: Capturing Raw Image

3. **Image Processing:**
    - **Region of Interest (ROI) Extraction:** The first detector scanned the entire video frame to identify the hand area, ensuring its precise cropping.
    - **Hand Landmark Detection:** The cropped area was processed using a second hand detector, mapping 21 key hand landmarks as shown in **Figure 6.2** which were instrumental in generating skeletonized hand representations, crucial for analyzing hand motion and positioning.
    - **Noise Removal:** The extracted ROI was transformed to grayscale, blurred using Gaussian blur, and put through thresholding to generate a high-contrast binary image that clearly isolated the hand gesture.
    - **Standardization:** Processed images were reconstructed to 400x400 pixels to ensure uniformity.
4. **Storage and Organization:** Images were stored in class-specific folders (e.g., “A,” “B”) for systematic organization, with a live count of images displayed to maintain an enhanced balanced dataset. This structure ensured efficient data access for training and testing processes.

Challenges Addressed:

- **Lighting and Background Variability:** Regulated preprocessing techniques helped improve recognition accuracy consistency.
- **Diversity in Hand Features:** Multiple users contributed to the dataset, incorporating variations in hand shapes, sizes, and skin tones for better generalization of the dataset.
- **Real-Time Processing:** The Jetson Nano's GPU acceleration ensured fast, efficient data collection and preprocessing, making the system reliable for real-world applications.

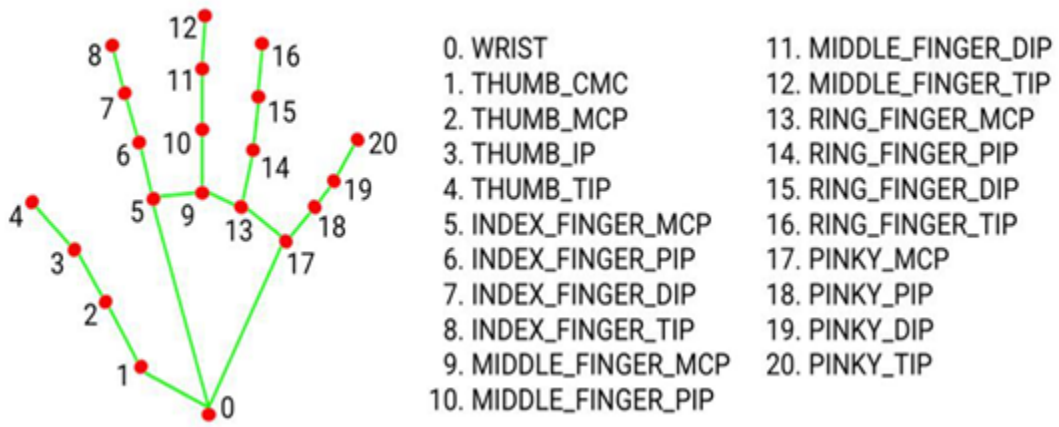

Fig. 6.2: Hand Landmark Points

6.2 Data Preprocessing

Efficient data preprocessing plays a crucial role in ensuring the improved performance of our Sign Language Recognition (SLR) system. The preprocessing pipeline enables data consistency, computational efficiency, and robust model generalization for real-time ASL recognition.The steps involved in data preprocessing process are presented in Fig. 6.3.

1. **Grayscale Conversion:** Instead of going forward with grayscale images, we transformed them into a three-channel RGB format. This maintains essential intensity values while ensuring compatibility with other deep learning architectures.
2. **Noise Reduction with Gaussian Blurring:** Applied Gaussian blur to smooth images, with reduction in background noise, and enhancement of the clarity of hand gestures.

3. **Thresholding for Contrast Enhancement**: Utilized adaptive thresholding to transform images into high-contrast binary format, ensuring clear demarkation between the hand gesture and background, even in uneven lighting conditions.
4. **Image Resizing:** All images were resized to 128x128 pixels, optimizing them for batch processing and inference on Jetson Nano. This ensures equalization across the dataset.
5. **Data Augmentation:** To simulate variations and increased model generalization, augmentation processes such as the following were applied in real-time:
    a. Shearing: Tilting the image to account for various hand orientations.
    b. Horizontal Flipping: Mirroring images to mimic left-hand gestures accurately.
    c. Scaling: Resizing gestures to simulate variable distances from the camera.

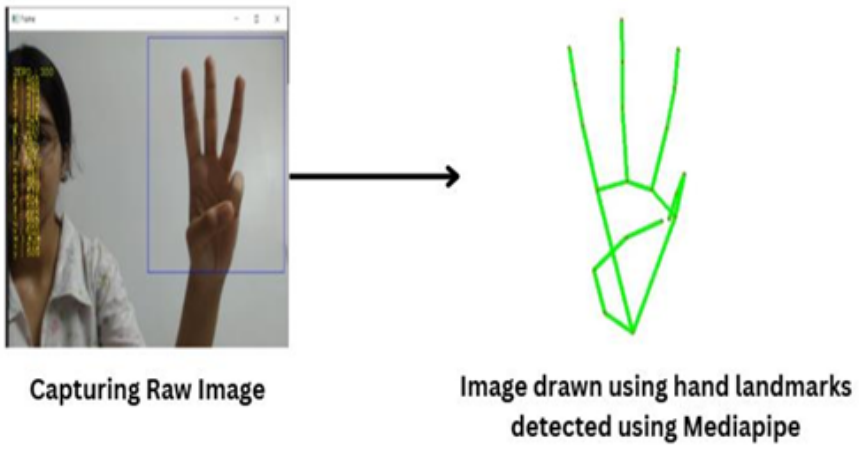

Fig. 6.3: Steps of Data Preprocessing

Optimized for Real-Time Performance

By leveraging the Jetson Nano's computational efficacy and custom data generators, preprocessing is optimized for real-time image processing and model training. This approach ensures that the dataset is processed, diverse, and suitable for deployment in real-world ASL recognition applications and instruments.

6.3. Model Training

Developing a robust model for ASL recognition utilized iterative improvements in model architecture and training methodologies. Table 6.1 compares the performance metrics of each model tested during the training process.

6.3.1 Model Selection

1. **InceptionV3:** The InceptionV3 architecture was initially executed for its multi-scale feature extraction capabilities, using convolutional kernels of varying sizes to capture both local and global patterns within a single image. This ability makes it compatible for recognizing complex hand gestures in sign language.

   Advantages of InceptionV3:

- **Efficient Feature Extraction**: Utilizes mixed-size convolutional kernels to increase recognition accuracy.
- **Reduced Computational Cost**: More optimized as compared to deeper architectures while maintaining a strong performance.
- **Transfer Learning Benefits**: Pre-trained ImageNet weights accelerated training and worked on improving generalization.

   Limitation: Despite its efficiency, InceptionV3 still presents deployment inefficacies on resource-constrained devices due to its moderate computational ask.

2. **MobileNetV2 (Final Model):** The demand for a lightweight yet efficient model capable of gesture recognition in real-time led to the adoption of MobileNetV2. This architecture leverages depthwise the facility of separable convolutions to significantly reduce computational model complexity while preserving higher accuracy.

   Key Features of MobileNetV2:

- **Inverted Residuals and Linear Bottlenecks**: Improves feature extraction while optimizing memory usage scenarios.

- **Depthwise Separable Convolutions**: converts standard convolutions into depthwise and pointwise operations to lower down the computational cost.
- **Low Latency**: Designed for real-time applications and efficient deployment on edge AI devices.

Benefits for ASL Recognition

- Delivers faster inference times compared to heavier architectures like InceptionV3 and Resnet.
- Compatible for deployment on resource-constrained devices, including smartphones and embedded systems.
- Maintains high accuracy while demanding minimal computational overhead.

Table 6.1: Comparison of Model Performance

Model	Training Accuracy	Testing Accuracy
MobileNetV2	99.8%	93%
InceptionV3	99.8%	88.52%

6.3.2 MobileNetV2 Architecture

Initially, **InceptionV3** was utilized for feature extraction due to its strong multi-scale representation abilities. However, to optimize computational efficiency for real-time ASL recognition, the system was fine-tuned by adopting **MobileNetV2** as the final model architecture.

MobileNetV2 is a 53-layer deep convolutional neural network designed for effective inference on resource-constrained devices. Pre-trained on over a million images from the ImageNet database, it can classify images upto 1,000 categories, ranging from objects such as keyboards and pencils to various animals. This extensive training ensures the model to learn rich and transferable feature representations. The network has an extensive input image size of 224×224 pixels and leverages depthwise separable convolutions to increase computational efficiency while

maintaining high accuracy. Fig. 6.4 illustrates the MobileNetV2 architecture, showcasing its optimized layers that allows for fast and accurate gesture classification in ASL.

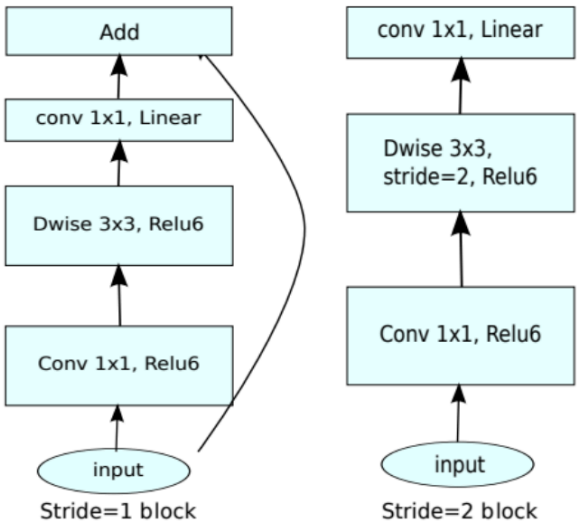

Fig. 6.4: MobileNetV2 architecture [27]

6.3.3 Customized Layers

The top layers of MobileNetV2 were refined with task-specific layers to adapt it for ASL recognition:

- **Global Average Pooling (GAP):** Reduction in dimensionality by averaging feature maps, reducing overfitting.
- **Dense Layers:** Two fully connected layers with ReLU activation to learn high-level representations of gestures.
- **Dropout Layer:** Applied with a rate of 0.5 to prevent overfitting by randomly deactivating neurons during training.
- **Softmax Output Layer:** Provided probabilistic outputs for 25 classes (letters).

Fig. 6.5 illustrates the complete model architecture, where MobileNetV2 is employed as the feature extractor (base model), and additional task-specific layers are appended to enhance its suitability for ASL gesture classification.

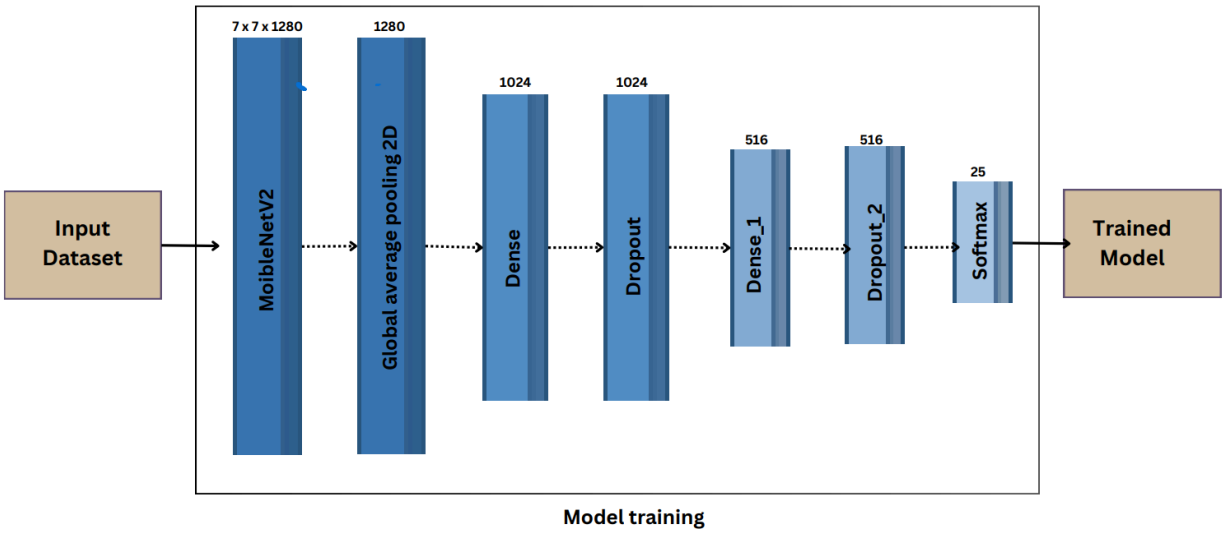

Fig. 6.5: Model Architecture and associated parameters

The architecture is designed for real-time inference on resource-constrained devices while maintaining high accuracy. Here's a detailed breakdown:

- **MobileNetV2 Base Model**: Pretrained on ImageNet, this model efficiently extracts high-level features from the input images using depthwise separable convolutions. It is kept frozen or partially trainable depending on the training strategy to reduce overfitting and speed up training.
- **Global Average Pooling Layer**: This layer reduces the dimensionality of the feature maps by computing the average value of each feature map. It helps in minimizing overfitting and ensures better generalization by flattening the spatial dimensions into a single vector.
- **Dense Layer (Dense_0)**: A fully connected layer that processes the pooled features and begins learning complex gesture-specific representations. Typically activated by ReLU for non-linearity.
- **Dropout Layer**: Applied after the first dense layer to randomly deactivate a portion of neurons during training (e.g., with a rate of 0.5), helping prevent overfitting.
- **Second Dense Layer (Dense_1)**: Further refines learned features for gesture classification.

- **Second Dropout Layer**: Reinforces regularization before the final classification.
- **Softmax Output Layer**: Outputs a probability distribution across the 25 gesture classes (24 ASL letters + 1 blank), allowing the model to make the final prediction based on the highest probability.

6.3.4 Optimization Process

1. **Hyperparameters:**
    - Optimizer: Adam, for its adaptive learning rate.
    - Learning Rate: Initially set to 0.001, dynamically reduced during training using callbacks.
    - Batch Size: 16, balancing memory usage and convergence speed.
    - Epochs: Trained for 15 epochs with early stopping to prevent overfitting.
2. **Loss Function:** Categorical Cross-Entropy: Used for multi-class classification.
3. **Callbacks Implemented:**
    - Early Stopping: Terminated training if the validation loss plateaued for five consecutive epochs.
    - ReduceLROnPlateau: Reduced the learning rate by a factor of 0.1 upon detecting performance plateaus.
4. **Validation:** Used 20% of the training dataset for real-time validation to monitor overfitting and model performance.

6.4.4 Performance Metrics

- Training Accuracy: 99.84%.
- Testing Accuracy: 93%.

6.4 Real-Time Prediction Pipeline

We aimed to process live video frames dynamically and produce accurate gesture recognition with minimal latency.

6.4.1 Steps in Real-Time Prediction

1. **Frame Capture:** Captures frames from the webcam at 30 fps using OpenCV. Dynamically detects the Region of Interest (ROI) containing the hand gesture.

2. **Frame Preprocessing:** Applies grayscale conversion, Gaussian blurring, and thresholding to prepare the frame for model inference.Resizes the ROI to 224x224 pixels.
3. **Model Inference:** Passes the preprocessed frame to the trained MobileNetV2 model for prediction. Outputs class probabilities, selecting the highest probability as the recognized letter.
4. **Post-Processing:** Recognized letters are buffered and combined into words dynamically. Spell-checking ensures word accuracy, with suggestions provided using the Hunspell library.

6.4.2 Performance Optimization

- **Latency:** Achieved an average processing time of 50–70 milliseconds per frame by utilizing GPU acceleration.
- **Error Handling:** Implemented fallback mechanisms to handle low-confidence predictions or missed frames.

6.5 Text- to- speech conversion and GUI Development

To enhance user interaction and accessibility, the Sign Language Recognition (SLR) system was equipped with a graphical user interface (GUI) using Tkinter on the Jetson Nano. The interface supports seamless real-time recognition of ASL gestures and delivers both visual and auditory feedback for an inclusive user experience. A user-friendly layout displays recognition results in real time, with integrated Text-to-Speech (TTS) functionality that supports three languages—Hindi, English, and Tamil—allowing users to select their preferred language for speech output. Fig. 6.6 illustrates the user interface and its functionality.

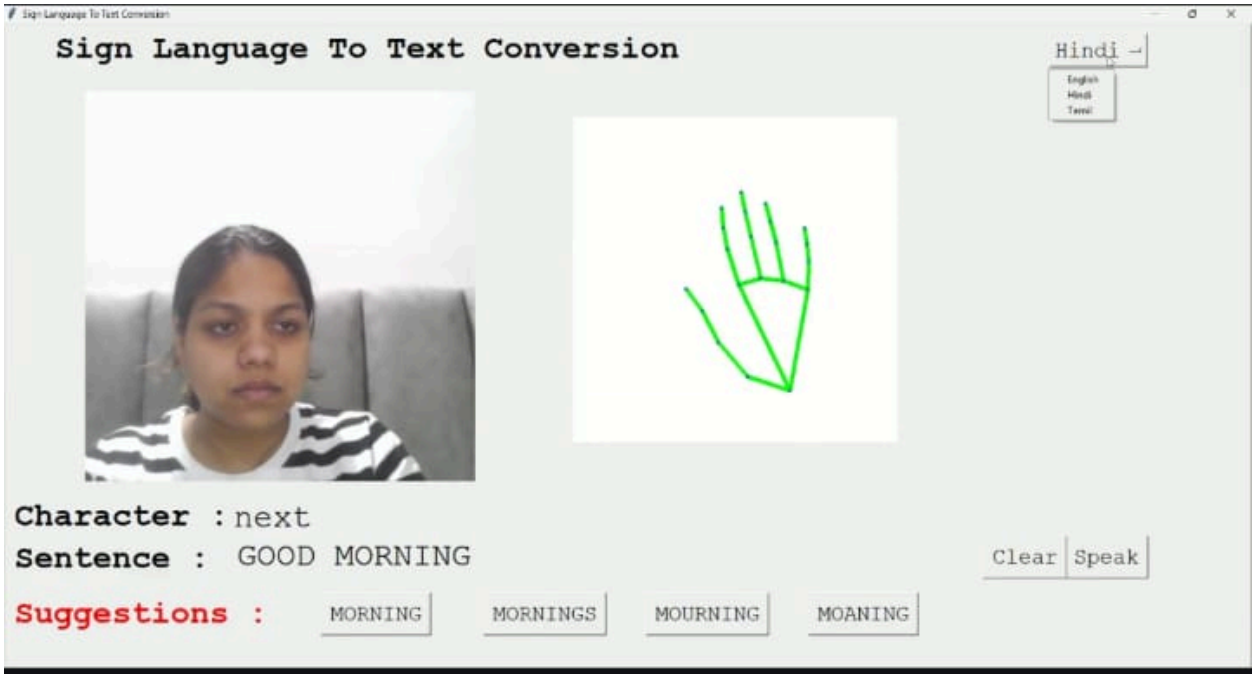

Fig. 6.6: Application GUI

Key features of the application include:

- **Live Gesture Recognition:** Displays a continuous video feed while simultaneously updating the recognized letter, word, and sentence.
- **Real-Time Text Generation:** Captured gestures are transformed into text, forming coherent words and sentences.
- **Interactive Controls:** Provides options to reset text, clear buffers, and refine predictions using the spell-checking feature.
- **Auditory Feedback (TTS Integration):** Recognized text is transformed into speech using a Text-to-Speech (TTS) module. A USB Sound Adapter enables seamless vocalization through Jetson Nano's integrated speakers, improving accessibility for end-users relying on auditory cues.

CHAPTER 7
APPLICATION DEPLOYMENT ON NVIDIA JETSON NANO

Overview: *This chapter outlines the deployment of the Sign Language Recognition (SLR) system on the NVIDIA Jetson Nano. It details the setup of the software environment, real-time input handling, gesture recognition, and multilingual text-to-speech output.*

The Sign Language Recognition (SLR) system was deployed on the **NVIDIA Jetson Nano**, a compact and power-efficient embedded platform, enabling real-time ASL gesture recognition with low latency. The deployment process was structured to ensure optimal performance, minimal resource consumption, and seamless user interaction. Figure 7.1 illustrates the **workflow of the deployed application**, highlighting the interaction between video capture, gesture recognition, text processing, and speech output.

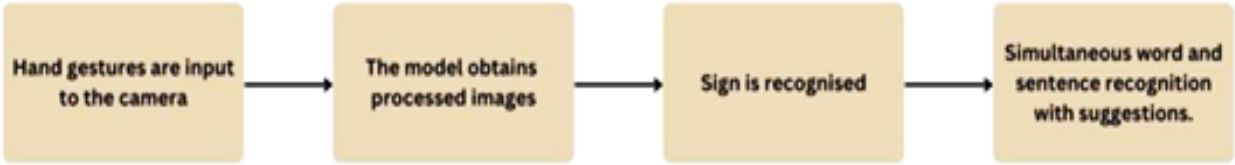

Fig. 7.1: Application Deployment Steps

7.1 Environment Setup

To prepare the Jetson Nano for the application, the following software environment was configured:

- **Python 3.9** was installed as the primary programming language.
- Required libraries included:
    - **TensorFlow** for deep learning model inference.
    - **OpenCV** for video capture and real-time frame processing.

    - **Tkinter** for designing the graphical user interface (GUI).
    - **pyttsx3** for offline Text-to-Speech (TTS) functionality.
    - **Hunspell** for real-time spell checking and word suggestion.
- The **USB camera** was connected and configured with the NVIDIA Jetson Nano to provide live video input of user hand gestures.

This setup ensured that all critical components of the system were integrated and operational on a low-power, edge-compatible device. Fig. 7.2 depicts how NVIDIA Jetson Nano was setup for the SLR application.

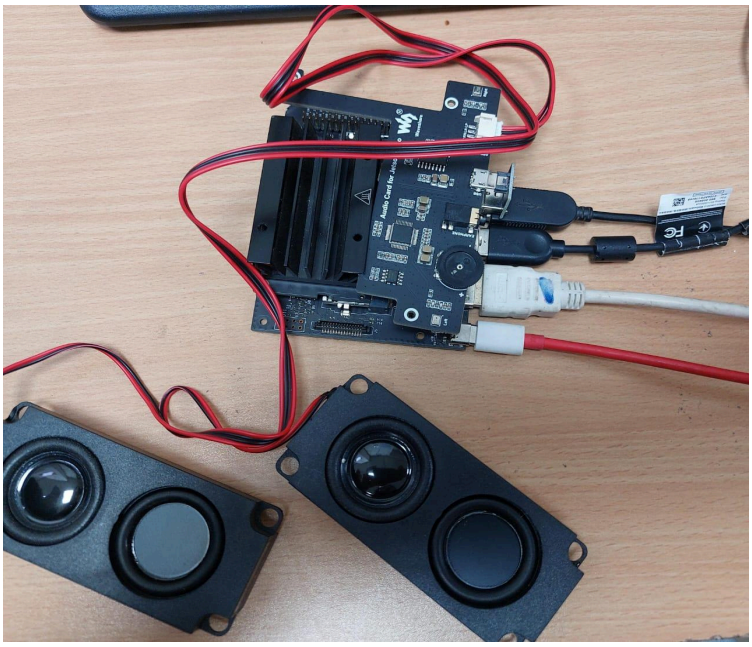

Fig. 7.2: NVIDIA Jetson Nano Setup with USB Speakers attached

7.2. Input Handling and Frame Processing

Once initialized, the application continuously captures video frames through the connected camera using OpenCV. The frames undergo several preprocessing steps to extract the **Region of Interest (ROI)**—primarily the user's hand gesture.

Preprocessing operations included:

- Grayscale conversion,

- Gaussian blurring,
- Adaptive thresholding,
- Cropping and resizing to 224×224 pixels (to match MobileNetV2 input requirements).

These processed frames are then passed into the fine-tuned MobileNetV2 model for classification of the ASL gesture.

7.3. Sign Recognition and Output Generation

Upon classification, the recognized letters are dynamically:

- Displayed on the GUI for user feedback,
- Buffered into coherent words and sentences,
- Validated and corrected using the Hunspell spell-checking engine.

To support non-signers and enhance accessibility, the system uses pyttsx3 to convert recognized text into audible speech, supporting multilingual output in English, Hindi, and Tamil. This multi-modal output ensures inclusive communication, bridging the gap between ASL users and listeners unfamiliar with sign language.

7.4. System Execution and Testing

The deployed application was tested in diverse real-world conditions to ensure robustness and reliability. The tests involved:

- Varying lighting environments,
- Different hand orientations and skin tones,
- Complex backgrounds.

Performance metrics included frame processing latency, gesture prediction accuracy, and speech output synchronization. The system consistently maintained real-time response with average processing latency between 50–70 milliseconds per frame, thanks to the Jetson Nano's GPU acceleration.

7.5. Deployment Issues and Challenges

Despite the successful deployment, several challenges were encountered during development:

- **Lighting Sensitivity**: The model performance slightly degraded under low-light or high-glare conditions, affecting the ROI extraction accuracy.
- **Background Noise**: Complex or cluttered backgrounds occasionally confused the classifier, necessitating further filtering or background subtraction techniques.
- **Gesture Similarity**: Certain hand gestures like 'M' vs. 'N' or 'U' vs. 'V' were harder to distinguish, causing occasional misclassifications.
- **Hardware Limitations**: Although the Jetson Nano provides GPU acceleration, running intensive inference and GUI operations simultaneously pushed the limits of its memory and processing capacity.
- **Speech Synchronization**: Delays in text-to-speech output were noticed under continuous gesture input, requiring optimization of the speech buffer and processing flow.

These insights helped refine the system and identify areas for future improvement and enhancement.

7.6. Feedback from Users

Strengths Identified:

- **High Recognition Accuracy:** Users found the system highly reliable in correctly identifying ASL fingerspelling gestures.

- **Real-Time Translation:** The quick response time between gesture input and output made interactions feel natural and smooth.

- **Multilingual Text-to-Speech:** Support for English, Hindi, and Tamil was praised for making the system accessible to a diverse user base.

- **User-Friendly Interface:** The graphical interface was appreciated for being simple, clear, and easy to navigate, even for first-time users.

Suggestions for Improvement:

- **Expansion of Gesture Range:** Users suggested extending the system to include dynamic signs for more comprehensive communication.

- **Enhanced Portability:** Some users recommended developing a mobile application or more compact wearable version for greater ease of use in daily life.

CHAPTER 8

SYSTEM TESTING AND PERFORMANCE METRICS

Testing is the most crucial phase, ensuring the robustness, accuracy, and usability of the system under real-world scenarios. The testing process involves verifying individual system components and their subsequent integration, simulating diverse conditions, and identifying potential problems to optimize system performance. Additionally, testing was conducted on Jetson Nano to ensure compatibility on an embedded edge device.

8.1 Testing Objectives

1. **Accuracy Testing:** To measure the precision of ASL letter recognition and word/sentence formulation under varying environmental and usage conditions.
2. **Performance Testing:** To evaluate the system's real-time responsiveness on the Jetson Nano, ensuring minimal latency during gesture recognition and maintaining high-yield performance under varying lighting, hand orientation, gesture speed, and background complexity scenarios.
3. **User Interaction Testing:** To verify the usability and functionality of the graphical user interface (GUI), and the efficacy of the real-time Text-to-Speech (TTS) output in English, Hindi, and Tamil, providing an inclusive multimodal experience for diverse users.

8.2 Test Cases and Results

Table 7.1 below summarizes the key test cases designated to evaluate the system:

Table 8.1: Key test cases

Test Case ID	Description	Expected Outcome	Result
TC1	Recognizing static ASL gestures under standard lighting	Recognize the correct letter with 90%+ accuracy	Pass

TC2	Dynamic text display in GUI	Update recognized letters, words, and sentences	Pass
TC3	Real-time responsiveness	Latency < 100 ms per frame	Pass
TC4	Gesture recognition in low-light conditions	Recognize letters with >70% accuracy	Pass (75%)
TC5	Recognition with cluttered/complex backgrounds	Maintain >80% accuracy	Partial Pass (78%)
TC6	Handling diverse hand sizes and orientations	Recognize gestures across multiple users	Pass
TC7	Robust spell-checking and word formation	Suggest and correct words accurately	Pass
TC8	User reset functionality in the GUI	Successfully clears buffer and resets system	Pass
TC9	Multilingual speech output via TTS	Audible output in selected language (English, Hindi, Tamil)	Pass
TC10	TTS and GUI integration on Jetson Nano	Real-time text and speech output without lag or crash	Pass

8.3 Testing Metrics

8.3.1 Accuracy Metrics

- Accuracy: Percentage of correctly recognized ASL letters during testing.
    - Training Accuracy: 99.8%
    - Testing Accuracy: 93%

8.3.2 Latency Metrics

- Average processing time per frame: 50–70 ms, measured from frame capture to prediction output

8.4 Observations from Testing

1. **Strengths:**
    - High recognition accuracy and low latency in controlled environments.
    - Real-time visual & audible feedback through GUI & TTS enhanced interactivity.
    - Choice of three languages (Hindi, English, Tamil) in speech output made the system more inclusive and user oriented.
    - Smooth operation on Jetson Nano confirmed its feasibility on low-powered edge devices.
    - Modular architecture enabled easy integration of new features.
2. **Weaknesses:**
    - Performance slightly degraded in complex backgrounds due to increased noise in ROI extraction.
    - Recognition accuracy dropped under low-light conditions, indicating a need for improved preprocessing techniques or model robustness.
3. **Limitations Identified:**
    - Dynamic gestures ('J' and 'Z') were not recognized, limiting the system's overall functionality.
    - Lack of context-aware grammar for sentence formation.

8.5 Improvements Post-Testing

On the basis of test results and user feedback, the following enhancements were included:

1. **Preprocessing Adjustments:** Adaptive thresholding was fine-tuned for improved performance under varying lighting conditions.
2. **Model Refinements:** Additional training with augmented low-light images improved performance in complex conditions.
3. **GUI and TTS Enhancements**:

- Included a "Suggestion" section and visual indicators for letter recognition confidence.
- A language selection option was added to the GUI, enabling users to choose between Hindi, English, and Tamil for speech output.

8.6 Testing Summary

The system successfully achieved its core objectives of real-time ASL recognition, multilingual speech output, as well as user-friendly operation on the Jetson Nano. Testing not only validated the effectiveness of the overall architecture but also guided the development of key enhancements. The integration of TTS in multiple languages significantly boosted accessibility, and performance optimization on the Jetson Nano proved the system's viability in real-world, resource-constrained deployments. Future work will focus on dynamic gesture support and context-aware sentence construction to further improve communication fidelity.

CHAPTER 9
RESULTS AND DISCUSSIONS

The proposed approach employs a transfer learning technique; it starts by using a pretrained MobileNetV2 model and then fine-tunes it using a self-captured dataset. Majority of the prior research relies on self-captured datasets that lack racial diversity, which impacts the model's predictions.

The MobileNetV2 model has been trained with 15,025 images from 25 classes with a batch size of 16. We aimed to assess the effectiveness of the MobileNetV2 model for recognition of letters and evaluate its potential for finding coherent text-based sentences through progressive word formation. It obtained a training and testing accuracy of 99.8% and 93% respectively. By pre-processing the data and adjusting the batch size and number of epochs, resulted in increased accuracy until the ideal parameters were found. Fig. 9.1 represents the model accuracy and Fig. 9.2 represents the loss while training for 15 epochs with a batch size of 16.

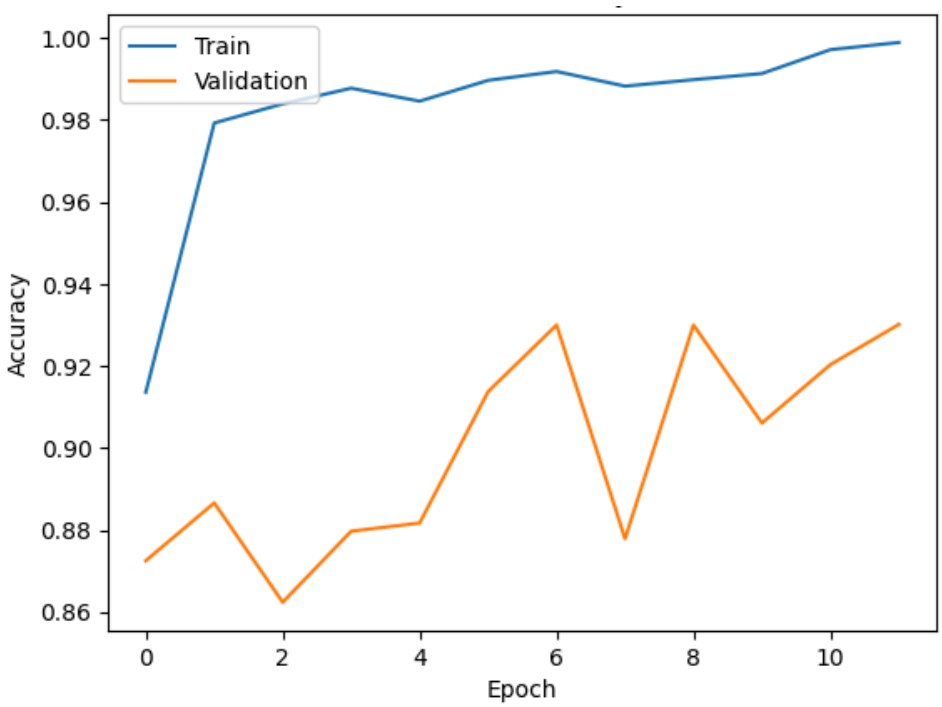

Fig. 9.1: Model Accuracy

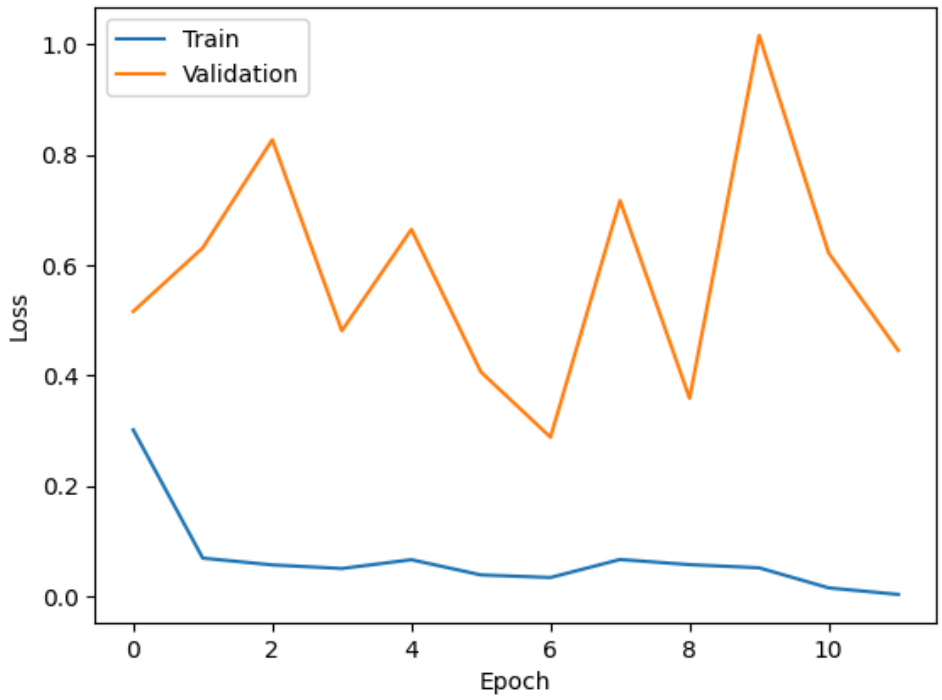

Fig. 9.2: Model Loss

Fig. 9.3 illustrates the confusion matrix generated for the trained MobileNetV2 model, providing a visual summary of its performance across all 26 classes (A–Z, excluding dynamic gestures 'J' and 'Z'). Each row of the matrix represents the actual class, while each column represents the predicted class. The diagonal elements indicate the number of instances where the predicted label matches the true label, thus reflecting correct classifications.

Higher values along the diagonal suggest that the model has effectively learned to recognize most of the ASL characters with minimal confusion. Off-diagonal elements, which represent misclassifications, are relatively sparse, indicating that the model performs well with limited confusion between similar-looking gestures. However, some overlapping features among certain characters—such as between 'M' and 'N', or 'U' and 'V'—may lead to occasional misclassification, as seen in a few non-diagonal cells.

This confusion matrix not only highlights the strengths of the model in distinguishing visually distinct signs but also pinpoints specific areas where further fine-tuning or additional data augmentation could help reduce misclassification rates.

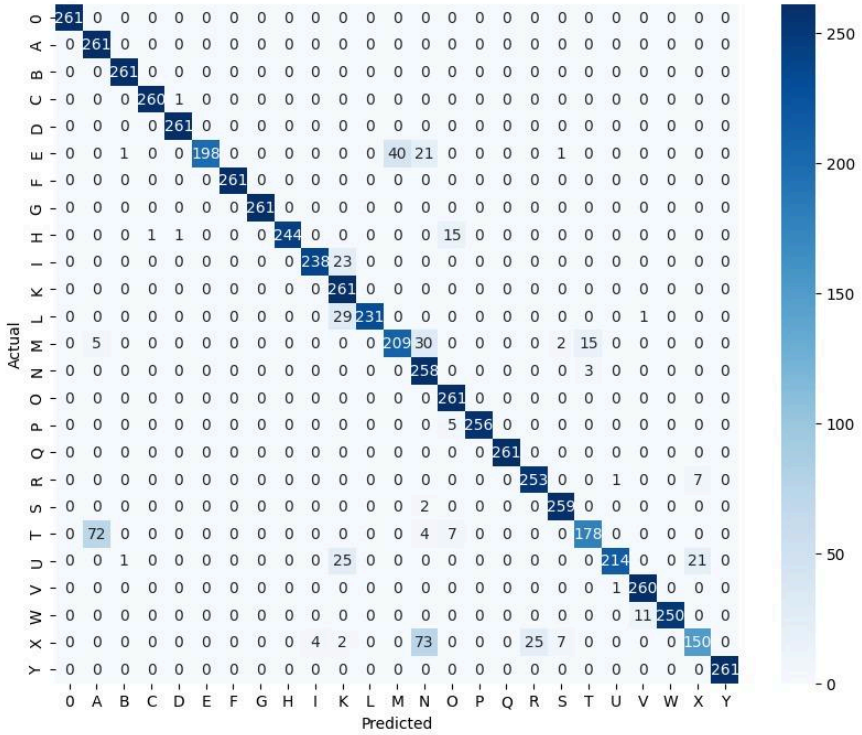

Fig. 9.3: Confusion matrix for Classification

The success of our approach can be attributed to several key factors:

1. **Comprehensive dataset:** We curated an extensive and diverse collection of 15,025 images, ensuring a wide range of variations in hand gestures, lighting conditions, and individual signing styles.
2. **Transfer learning:** By leveraging the power of pre-trained architectures and fine-tuning them on our self-curated dataset, we capitalized on the power of transfer learning to achieve competitive and high-end performance.
3. **Text-to-speech integration**: The incorporation of a text-to-speech module transforms the recognized text into speech, further bridging the communication gap between signers and non-signers.

Table 8.1 presents a comparative analysis between the proposed system and several notable studies in the domain of sign language fingerspelling recognition. This comparison highlights key differences in dataset characteristics, algorithmic approaches, and achieved accuracies.

Table 9.1 Comparison of previous and proposed work

Ref	Sign Language	Dataset	Sample Size	Algorithm	Accuracy
[29]	Spanish	Self Captured	28,862	CNN-Resnet	79.96%
[30]	American	Self Captured	2080	RNN	93.4%
[31]	American	NCSLGR	3085	KNN	91.27%
[32]	American	ChicagoFSWild	7304	Hybrid CTC-Attention Model	71.7%
[33]	British	Self Captured	1000	HMM	98.9%
Proposed	**American**	**Self Captured**	**15,025**	**CNN-MobileNet**	**93%**

The table underscores the diversity of sign languages (Spanish, American, British) and datasets used across various studies. Notably, self-captured datasets were common, reflecting the need for tailored data in gesture-based tasks. Sample sizes vary significantly—from smaller datasets like 1,000 samples in [33] to large-scale collections like 28,862 samples in [29]—indicating differing levels of model generalization and robustness.

In contrast, the proposed system uses a CNN-MobileNet architecture, tailored for real-time deployment on edge devices like the Jetson Nano. With a self-captured American Sign Language dataset of 15,025 samples, the model achieves a competitive accuracy of 93%, combining both computational efficiency and strong generalization.

This comparison demonstrates that the proposed model effectively balances performance and resource efficiency, making it well-suited for real-world, real-time applications in sign language recognition.

CHAPTER 10
CONCLUSION & FUTURE WORK

This research presents a novel approach to American Sign Language (ASL) fingerspelling recognition using deep learning techniques, contributing to the development of a robust system capable of classifying 25 distinct ASL gestures with impressive accuracy. By integrating a text-to-speech module, the system goes beyond mere gesture recognition, offering a more holistic solution for sign language communication. The deployment of the system on the NVIDIA Jetson Nano enables efficient real-time performance, taking full advantage of its computational power to run complex models while ensuring low latency. The addition of a USB camera, speakers, and an audio card further enhances the system's functionality, enabling the transformation of recognized gestures into spoken words in real-time.

The system achieved a training accuracy of 99.8% and a testing accuracy of 93%, demonstrating the effectiveness of the proposed approach and the advantages of transfer learning in gesture recognition. These results underscore the potential of the system to bridge the communication gap between ASL users and non-signers, enabling both real-time text and speech output. This multi-modal capability is a step toward greater social inclusion and accessibility for individuals who rely on sign language as their primary mode of communication.

Despite the promising outcomes, there are several areas for future improvement and expansion:

1. **Expanded Gesture Recognition:** Future work could extend the system's capabilities to recognize a broader range of ASL signs, including dynamic gestures and complete signs, moving beyond fingerspelling to more comprehensive language translation.
2. **Real-Time Performance Optimization:** While the system performs well in real-time, further optimizations in processing speed and resource efficiency could allow for even smoother performance, especially when adapting the system for use in mobile or wearable devices.

3. **Multi-Modal Integration:** Incorporating additional modalities, such as body posture and facial expressions, could enhance the nuance and accuracy of sign language interpretation, providing a more complete and context-aware translation.

By exploring these research avenues, we can build upon the foundation laid in this study to create more accurate, comprehensive, and accessible systems for ASL recognition, translation, and vocalization. These advancements hold the potential to greatly improve communication for the deaf and hard-of-hearing community, leading to a more inclusive society where language barriers are continuously reduced. Ultimately, the integration of visual, textual, and auditory communication modes represents a significant step forward in developing inclusive communication technologies.

References

[1] V. Bheda and D. Radpour, “Using Deep Convolutional Networks for Gesture Recognition in American Sign Language,” *CoRR*, vol. abs/1710.06836, 2017, [Online]. Available: http://arxiv.org/abs/1710.06836

[2] L. C. L. D. S. S. V. G. I. N. Fei Xu1, *A Comparative Study of Video-based Human Representations for American Sign Language Alphabet Generation*. IEEE, 2024.

[3] B. Shi, D. Brentari, G. Shakhnarovich, and K. Livescu, “Fingerspelling Detection in American Sign Language.”

[4] M. D. Nareshkumar and B. Jaison, “A Light-Weight Deep Learning-Based Architecture for Sign Language Classification,” *Intelligent Automation and Soft Computing*, vol. 35, no. 3, pp. 3501–3515, 2023, doi: 10.32604/iasc.2023.027848.

[5] R. Rastgoo, K. Kiani, and S. Escalera, “Sign Language Recognition: A Deep Survey,” *Expert Syst Appl*, vol. 164, p. 113794, Sep. 2020, doi: 10.1016/j.eswa.2020.113794.

[6] N. Adaloglou *et al.*, “A Comprehensive Study on Sign Language Recognition Methods,” *CoRR*, vol. abs/2007.12530, 2020, [Online]. Available: https://arxiv.org/abs/2007.12530

[7] H. Zhou, W. Zhou, Y. Zhou, and H. Li, “Spatial-Temporal Multi-Cue Network for Continuous Sign Language Recognition.” [Online]. Available: www.aaai.org

[8] N. Camgoz, O. Koller, S. Hadfield, and R. Bowden, “Multi-channel Transformers for Multi-articulatory Sign Language Translation,” 2020, pp. 301–319. doi: 10.1007/978-3-030-66823-5_18.

[9] K. Yin, A. Moryossef, J. Hochgesang, Y. Goldberg, and M. Alikhani, “Including Signed Languages in Natural Language Processing,” *CoRR*, vol. abs/2105.05222, 2021, [Online]. Available: https://arxiv.org/abs/2105.05222

[10] O. M. Sincan and H. Y. Keles, “AUTSL: A Large Scale Multi-modal Turkish Sign Language Dataset and Baseline Methods,” *CoRR*, vol. abs/2008.00932, 2020, [Online]. Available: https://arxiv.org/abs/2008.00932

[11] D. Li, C. R. Opazo, X. Yu, and H. Li, “Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison,” *CoRR*, vol. abs/1910.11006, 2019, [Online]. Available: http://arxiv.org/abs/1910.11006

[12] Y. C. Bilge, N. Ikizler-Cinbis, and R. G. Cinbis, “Zero-Shot Sign Language Recognition: Can Textual Data Uncover Sign Languages?,” *CoRR*, vol. abs/1907.10292, 2019, [Online]. Available: http://arxiv.org/abs/1907.10292

[13] D. Metaxas, M. Dilsizian, and C. Neidle, “Linguistically-driven Framework for Computationally Efficient and Scalable Sign Recognition,” Sep. 2018.

[14] D. Bragg *et al.*, “Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective,” Sep. 2019, pp. 16–31. doi: 10.1145/3308561.3353774.

[15] N. C. Camgöz, O. Koller, S. Hadfield, and R. Bowden, “Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation,” *CoRR*, vol. abs/2003.13830, 2020, [Online]. Available: https://arxiv.org/abs/2003.13830

[16] N. Aloysius and P. Nedungadi, “Continuous Sign Language Recognition with Adapted Conformer via Unsupervised Pretraining.”

[17] O. V Murthy and R. Goecke, “The Influence of Temporal Information on Human Action Recognition with Large Number of Classes,” *2014 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2014*, Sep. 2015, doi: 10.1109/DICTA.2014.7008131.

[18] H. Reza, V. Joze, M. Redmond, and O. Koller, “MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language.” [Online]. Available: https://www.microsoft.com/en-us/research/project/ms-asl/

[19] K. Eepuri, P. V. V Kishore, T. Maddala, D. Anil Kumar, and S. S, “3D Sign Language Recognition with Angular Velocity Maps and Connived Feature ResNet,” *IEEE Signal Process Lett*, vol. PP, p. 1, Sep. 2018, doi: 10.1109/LSP.2018.2877891.

[20] R. Kumar, S. K. Singh, A. Bajpai, and A. Sinha, “Mediapipe and CNNs for Real-Time ASL Gesture Recognition.”

[21] A. Sharma and V. Sharma, “ASL Fingerspelling Recognition Using Hybrid Deep Learning Architecture,” *International Research Journal of Engineering and Technology*, 2023, [Online]. Available: www.irjet.net

[22] R. A. Kadhim and M. Khamees, “A real-time american sign language recognition system using convolutional neural network for real datasets,” *TEM Journal*, vol. 9, no. 3, pp. 937–943, Aug. 2020, doi: 10.18421/TEM93-14.

[23] B. Kaur, A. Chaudhary, S. Bano, Yashmita, S. R. N. Reddy, and R. Anand, “Fostering inclusivity through effective communication: Real-time sign language to speech conversion system for the deaf and hard-of-hearing community,” *Multimed Tools Appl*, vol. 83, no. 15, pp. 45859–45880, May 2024, doi: 10.1007/s11042-023-17372-9.

[24] “Jetson modules, support, ecosystem, and lineup. NVIDIA Developer (2023).” [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano

[25] “Raspberry Pi 4 (2020).” [Online]. Available: https://www.raspberrypi.com/documentation/

[26] “Coral, Dev Board, Google (2020).” [Online]. Available: https://coral.ai/products/dev-board/

[27] “Jetson Xavier NX (2020).” [Online]. Available: https://developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit

[28] “Jetson-AGX-Xavier .” [Online]. Available: https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/jetson-agx-xavier

[29] F. Morillas-Espejo and E. Martinez-Martin, “A real-time platform for Spanish Sign Language interpretation,” *Neural Comput Appl*, 2024, doi: 10.1007/s00521-024-10776-0.

[30] Y. Gu, Sherrine, W. Wei, X. Li, J. Yuan, and M. Todoh, “American Sign Language Alphabet Recognition Using Inertial Motion Capture System with Deep Learning,” *Inventions*, vol. 7, no. 4, Dec. 2022, doi: 10.3390/inventions7040112.

[31] P. Yanovich, C. Neidle, and D. Metaxas, “Detection of Major ASL Sign Types in Continuous Signing for ASL Recognition Detection of major ASL sign types in continuous signing for ASL recognition Detection of Major ASL Sign Types in Continuous Signing for ASL Recognition,” 2016. [Online]. Available: https://hdl.handle.net/2144/27492

[32] P. Pannattee, W. Kumwilaisak, C. Hansakunbuntheung, N. Thatphithakkul, and C.-C. Kuo, “American Sign Language Fingerspelling Recognition in the Wild with Spatio Temporal Feature Extraction and Multi-task Learning,” Jan. 2022.

[33] Liwicki Stephan and Everingha Mark, *Automatic Recognition of Fingerspelled Words in British Sign Language*. IEEE, 2009.

APPENDIX

User Manual

Setup Instructions

1. Clone the repository

```
git clone https://github.com/bh-g/Sign-Language-Recognition.git
```

2. Create and activate a virtual environment (optional but recommended)

```
python -m venv venv
source venv/bin/activate  # for Mac/Linux
venv\Scripts\activate     # for Windows
```

3. Install the required dependencies

```
pip install -r requirements.txt
```

4. Run the project

```
Python application.py
```

Hardware Setup for NVIDIA Jetson Nano

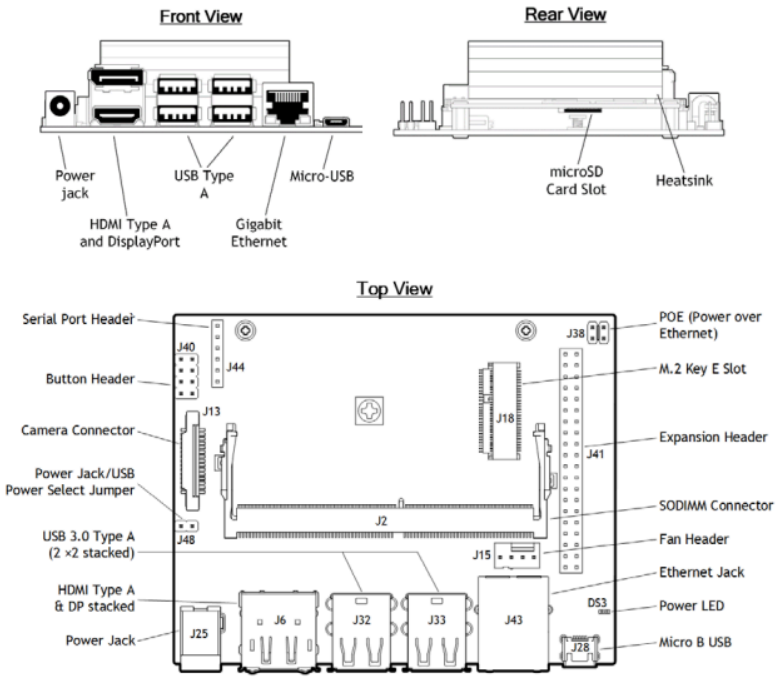

Fig 10: Developer Kit Module and Carrier Board

1. Jetson Nano Board

- Use a Jetson Nano Developer Kit.
- Insert a microSD card (at least 32GB, Class 10/UHS-1) flashed with JetPack (Jetson's Ubuntu-based OS).

2. Power Supply

- Recommended: 5V 4A DC Barrel Jack power supply.
- Alternate: 5V 2.5A micro-USB (only for lightweight tasks).
  Important: Insufficient power may cause random reboots or throttling!

3. Peripherals

- Keyboard and Mouse via USB.
- Monitor connected through HDMI or DisplayPort.
- Internet Connection:
  - Wired Ethernet is preferred (faster and more stable).
  - USB Wi-Fi adapter (make sure it's compatible with Ubuntu 18.04/20.04).
- Audio Output (Speakers): USB Speakers (plug into USB port).
- Camera: USB Webcam (plug into USB port).

Setup:

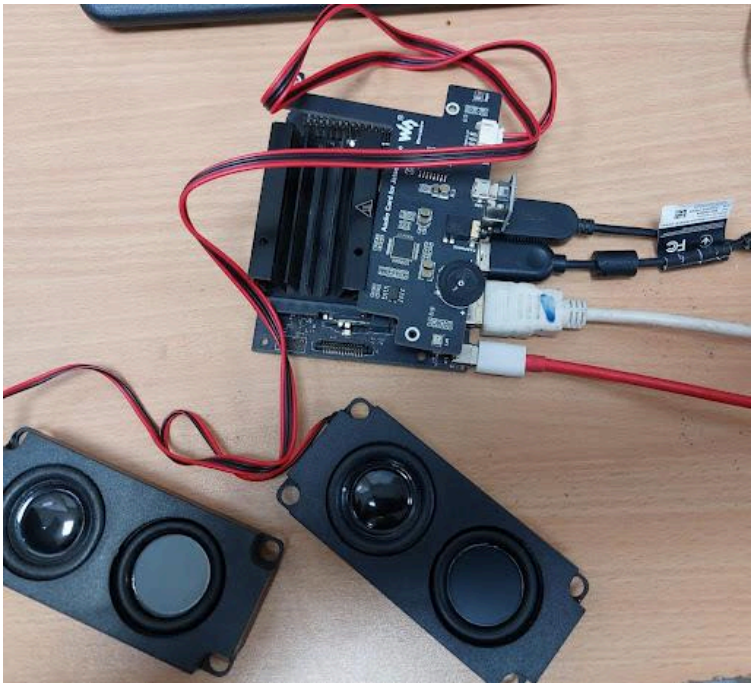

Fig 11: Setup for Jetson Nano

Final Checklist Before Running

- Jetson Nano powered and booted up.
- Monitor, keyboard, mouse connected.
- Speakers connected and tested.
- Network (Ethernet or Wi-Fi) connected.
- Camera or other USB devices attached (if required).
- Software dependencies installed.

For detailed instructions, please refer to the official Jetson Nano Developer Kit User Guide.

PUBLICATION DETAILS

1. M. Chandna, A. Dasgupta, B. Gupta, R.M., S. R. N. Reddy, and R. Anand, "Bridging the gap: Real-Time ASL Fingerspelling to Sentence Translation," *Proceedings of the 6th International Conference on Intelligence and Speech Technology (AIST-2024)*. (Accepted and presented; publication in proceedings pending).

Abstract:

Sign language represents an essential communication tool that benefits the Deaf and hard-of-hearing community, yet the lack of effective communication with non-signers often leads to social isolation. This research emphasizes on the development of an application using Tkinter for recognizing words in American Sign Language (ASL) through fingerspelling and turning them into sentences. It presents a novel approach to converting American sign language into text in real time, utilizing cutting-edge computer vision and deep learning techniques. The proposed method uses a webcam-based self-made dataset and a custom model with the InceptionV3 architecture as the basic model to categorize the letters via transfer learning. The recognized letters were then progressively combined to form words, which were further computed into meaningful sentences. Our approach focuses on improving recognition accuracy while maintaining real-time processing for minimizing the communication gap and contributing to promote social inclusivity of the Deaf community.

Keywords: Sign Language Recognition, Deep Learning, InceptionV3

Proof of presentation:

6TH INTERNATIONAL CONFERENCE ON
ARTIFICIAL INTELLIGENCE & SPEECH TECHNOLOGY
AIST-2024

CERTIFICATE OF APPRECIATION

This certificate is awarded to

Manya Chandna from IGDTUW

for presenting a Research Paper titled

Bridging the Gap: Real-Time ASL Fingerspelling to Sentence Translation

in the 6thAIST-2024 jointly organized by Centre of Excellence - AI and Department of IT, Indira Gandhi Delhi Technical University for Women (IGDTUW), Delhi held on 13th- 14th November, 2024.

Financial Sponsor: Anusandhan National Research Foundation

Publication Partner: Springer CCIS

Prof. Arun Sharma
Organizing Chair - AIST 2024
Dean Academic Affairs, IGDTUW

Dr. Prof. Ranjana Jha,
General Chair, AIST-2024
Vice-Chancellor, IGDTUW

6TH INTERNATIONAL CONFERENCE ON
ARTIFICIAL INTELLIGENCE & SPEECH TECHNOLOGY
AIST-2024

BEST PAPER AWARD

The certificate of Best Paper is awarded to

Manya Chandna from IGDTUW, Delhi

for the paper titled Real-Time ASL Fingerspelling to Sentence Translation

in the 6thAIST-2024 jointly organized by Centre of Excellence - AI and Department of IT, Indira Gandhi Delhi Technical University for Women (IGDTUW), Delhi held on 13th- 14th November, 2024.

Financial Sponsor: Anusandhan National Research Foundation

Publication Partner: Springer CCIS

2. M. Chandna, A. Dasgupta, B. Gupta, R. M., S. R. N. Reddy, and R. Anand, "Real-time sign language recognition using deep learning on edge devices: A Jetson Nano implementation," *Proceedings of the 5th International Conference on Emerging Trends and Technologies on Intelligent Systems (ETTIS-2025)*. (Accepted and presented; publication in proceedings pending).

Abstract:

Sign language serves as an integral communication medium for the hard-of-hearing and deaf community. However, the lack of efficient recognition systems often leads to barriers in interaction between signers and non-signers. This study presents a real-time Sign Language Recognition System (SLR) system implemented on the Jetson Nano, leveraging its GPU-accelerated deep learning capabilities for high-speed and accurate recognition. The system uses transfer learning approach with the pre-trained InceptionV3 architecture as the base model to recognize American Sign Language (ASL) fingerspelling and progres sively form sentences. To further enhance the model performance, we created a custom dataset of 12,025 images spread across 25 classes, i.e., 24 ASL letters and one blank class. Model refinement occurs over 12 epochs, each with a batch size of 16, culminating in training and testing accuracies of 99.84% and 88.52% re spectively. By optimizing deep learning inference on the Jetson Nano, this work ensures real-time processing on edge devices, advancing accessibility and pro moting social inclusivity for the deaf community.

Keywords: Sign Language Recognition, Jetson Nano, Deep Learning

Proof of presentation:

Ref. No. :1612/333268/ETTIS25/PP/2024

सी डैक
CDAC

A Scientific Society of the Ministry of Electronics and Information Technology, Government of India

5th International Conference on
Emerging Trends and Technologies on Intelligent Systems
(ETTIS-2025)
19th – 21st March, 2025

Certificate

This is to certify that Prof. / Dr. / Mr. / Ms. Bhumika Gupta *has presented/contributed the paper titled* Real-Time Sign Language Recognition Using Deep Learning on Edge Devices: A Jetson Nano Implementation *in International Conference on Emerging Trends and Technologies on Intelligent Systems (ETTIS-2025) in association with the Petroleum-Gas University of Ploiesti, Romania, the University of Haute-Alsace, France and Northeastern University, USA.*

(Dr. Kriti Saroha)
Scientist 'E' &
Program Chair (ETTIS-2025)

(Dr. Kalpana Johari)
Scientist 'F' &
Principal General Chair (ETTIS-2025)

(Vivek Khaneja)
Executive Director &
Patron (ETTIS-2025)

UNIVERSITÉ HAUTE-ALSACE
MGEN
Northeastern University College of Engineering
Springer

3. M. Chandna, A. Dasgupta, B. Gupta, R. M., S. R. N. Reddy, and R. Anand, "Edge AI for real-time sign language to speech conversion: Implementing deep learning on Jetson Nano," *Journal of Psycholinguistic Research*, 2025.- under review(Communicated)

Abstract:

Seamless communication is paramount in empowering the deaf and hard-of-hearing individuals, with sign language acting as a vital conduit for expres-sion. To enhance their societal interactions with non- signers, this study pio-neers the development of an advanced application using Tkinter for recog-nizing words in American Sign Language (ASL) through fingerspelling and subsequently forming sentences leveraging the computational prowess of Jetson Nano. The seamless integration of our sign language recognition ca-pabilities with text-to-speech conversion completes the communication loop. Signed expressions are converted to audible speech paving the way for their application in real time communication aids and ASL education tools. The proposed method utilizes a webcam-based self-made training dataset of 15,025 meticulously categorized images spanning 25 classes, which are thereafter preprocessed, and a custom model with MobileNetV2 as the foun-dational architecture to categorize the letters via transfer learning. The recog-nized alphabets were then progressively pooled to form words, which were further computed into meaningful sentences, culminating in real-time speech synthesis—all orchestrated through Jetson Nano. The system's ability to expedite real-time communication between non-signers and signers represents an essential step towards breaking down communication barriers and foster-ing profound understanding between diverse groups.

Keywords: Sign Language Recognition, Deep Learning, MobileNetV2, NVIDIA Jetson Nano

Proof-

Journal of Psycholinguistic Research - Submission Notification to co-author - [EMID:2937b1663bc45e72] External Inbox ×

Journal of Psycholinguistic Research (JOPR) <em@editorialmanager.com> 12:05 PM (4 minutes ago)
to me

Re: "Edge AI for Real-Time Sign Language to Speech Conversion: Implementing Deep Learning on Jetson Nano"

Full author list: Manya Chandna; Abantika Dasgupta; Bhumika Gupta; Ravinder M.; S.R.N. Reddy; Rishika Anand

Dear Ms. Chandna,

We have received the submission entitled: "Edge AI for Real-Time Sign Language to Speech Conversion: Implementing Deep Learning on Jetson Nano" for possible publication in Journal of Psycholinguistic Research, and you are listed as one of the co-authors.

The manuscript has been submitted to the journal by Dr. Ms. Rishika Anand who will be able to track the status of the paper through his/her login.

If you have any objections, please contact the editorial office as soon as possible. If we do not hear back from you, we will assume you agree with your co-authorship.

Thank you very much.

With kind regards,
Springer Journals Editorial Office
Journal of Psycholinguistic Research

PlagiarismReport

turnitin Page 2 of 68 - Integrity Overview Submission ID trn:oid:::1:3228398689

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

- 87 Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- 1 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 1 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
7% Publications
3% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 68 - Integrity Overview Submission ID trn:oid:::1:3228398689