IDENTIFICATION OF INDIAN SIGN LANGUAGE

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

ODRIN RODRIGUES

SHIVAM YADAV

KAUSTUBH DESLE

ANUSHKA SHINDE

UNDER THE GUIDANCE OF

PROF. MEENA UGALE

(DEPARTMENT OF INFORMATION TECHNOLOGY)

INFORMATION TECHNOLOGY DEPARTMENT

XAVIER INSTITUTE OF ENGINEERING

UNIVERSITY OF MUMBAI

2022 - 2023

XAVIER INSTITUTE OF ENGINEERING

Institute Vision

To nurture the joy of excellence in a world of high technology.

Institute Mission

To strive to match global standards in technical education by interaction with industry, continuous staff training and development of quality of life.

Xavier Institute of Engineering
Mahim, Mumbai 400016
Department of Information Technology
(Affiliated to University of Mumbai)

Department Vision

To nurture the joy of excellence in the world of Information Technology.

Department Mission

M1: To develop the critical thinking ability of students by promoting interactive learning.

M2: To bridge the gap between industry and institute and give students the kind of exposure to the industrial requirements in current trends of developing technology.

M3: To promote learning and research methods and make them excel in the field of their study by becoming responsible while dealing with social concerns.

M4: To encourage students to pursue higher studies and provide them awareness on various career opportunities that are available.

Xavier Institute of Engineering
Mahim, Mumbai 400016
Department of Information Technology
(Affiliated to University of Mumbai)

Program Educational Objectives (PEOs)

Information Technology Engineering Graduates will be

PEO1: Employed as IT professionals, and shall engage themselves in learning, understanding, and applying newly developed ideas and technologies as their field of study evolves.

PEO2: Competent to use the learnt knowledge successfully in the diversified sectors of industry, academia, research and work effectively in a multidisciplinary environment.

PEO3: Aware of professional ethics and create a sense of social responsibility in building the nation/society.

Program Specific Outcomes (PSOs)

PSO1: Demonstrate the ability to analyze and visualize the business domain and formulate appropriate information technology solutions.

PSO2: Apply various technologies like Intelligent Systems, Data Mining, IOT, Cloud and Analytics, Computer and Network Security etc. for innovative solutions to real time problems.

Xavier Institute of Engineering
Mahim, Mumbai 400016
Department of Information Technology
(Affiliated to University of Mumbai)

Program Outcomes (POs)

Engineering Graduates will be able to

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3: Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions for complex problems.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

XAVIER INSTITUTE OF ENGINEERING
MAHIM CAUSEWAY, MAHIM,MUMBAI - 400016.

CERTIFICATE

This is to certify that

ODRIN RODRIGUES	XIE IT 201903039
SHIVAM YADAV	XIE IT 201903054
KAUSTUBH DESLE	XIE IT 20180307
ANUSHKA SHINDE	XIE IT 2020032006

Have satisfactorily carried out the MAJOR PROJECT work titled "***IDENTIFICATION OF INDIAN SIGN LANGUAGE*** " in partial fulfillment of the degree of Bachelor of Engineering as laid down by the University of Mumbai during the academic year 2022-2023.

Prof. Meena Ugale
Supervisor/Project Guide

Dr. Jaychand Upadhyay
Head Of Department

Dr. Y. D. Venkatesh
Principal

PROJECT REPORT APPROVAL FOR B.E.

This project entitled

“IDENTIFICATION OF INDIAN SIGN LANGUAGE”

By

Name	Roll No.
ODRIN RODRIGUES	(XIE IT 201903039)
SHIVAM YADAV	(XIE IT 201903054)
KAUSTUBH DESLE	(XIE IT 20180307)
ANUSHKA SHINDE	(XIE IT 2020032006)

is approved for the degree of BACHELOR OF ENGINEERING.

Examiners

1. ________________

2. ________________

Date: 02/05/2023

Place: MAHIM, MUMBAI

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that I have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which thus have not been properly cited or from whom proper permission have not been taken when needed.

Name		**Signature**
ODRIN RODRIGUES	(XIEIT201903039)	
SHIVAM YADAV	(XIEIT201903054)	
KAUSTUBH DESLE	(XIEIT20180307)	
ANUSHKA SHINDE	(XIEIT2020032006)	

Date: 02/05/2023

ABSTRACT

Communication barriers can arise between deaf and normal individuals due to the language difference. Sign language is the primary mode of communication for deaf individuals, but normal people often have difficulty understanding their hand signs. Even for those who know sign language, effective communication requires a translator who also knows sign language. This can create difficulties and miscommunication, leading to frustration and isolation.

Addressing this issue, this project proposes a system that utilizes deep learning and computer vision techniques to identify sign language and generate sign animations based on text input using Blender. The system's sign language identification component employs a CNN-based model, achieving an accuracy of 98-99% on the test set, while transfer learning (KNN) is used for real-time video frame recognition. This approach can potentially bridge communication gaps between hearing and deaf individuals and enhance accessibility for the deaf community.

The proposed system's accuracy and features can greatly benefit communication between deaf and normal individuals. The CNN-based model's high accuracy ensures reliable identification of sign language, while the transfer learning (KNN) algorithm allows real-time recognition of sign language classes in video frames. Moreover, the system features an intuitive user interface, facilitating text input and sign animation playback. The system can be used for educational resources, entertainment, and virtual communication platforms, providing a means for effective communication between deaf and normal individuals.

Table of Contents

List of Figures

List of Tables

Acknowledgement

We would like to thank Fr. Dr. John Rose S.J (Director of XIE) for providing us with such an environment so as to achieve goals of our project and supporting us constantly.

We express our sincere gratitude to our Honorable Principal Dr. Y.D.Venkatesh for encouragement and facilities provided to us.

We would like to place on record our deep sense of gratitude to Dr.Jaychand Upadhyay, Head of Deptartment Of Information Technology, Xavier Institute of Engineering, Mahim, Mumbai, for his generous guidance help and useful suggestions.

With deep sense of gratitude we acknowledge the guidance of our project guide Prof. Meena Ugale. The time-to-time assistance and encouragement by her has played an important role in the development of our project.

We would also like to thank our entire Information Technology staff who have willingly cooperated with us in resolving our queries and providing us all the required facilities on time.

ODRIN RODRIGUES (XIEIT201903039)

ANUSHKA SHINDE (XIEIT2020032006)

KAUSTUBH DESLE (XIEIT20180307)

SHIVAM YADAV (XIEIT201903054)

Xavier Institute of Engineering
Mahim, Mumbai 400016
Department of Information Technology
(Affiliated to University of Mumbai)

CLASS: BE
SEM: VII/VIII
COURSE CODE: ITM701/ITM801
COURSE NAME: Major Project - VII/VIII1
AY: 2022-23

COURSE OBJECTIVE
This course aims:

CO No.	**COURSE OUTCOME**
CO1	Identify problems based on societal/research needs and apply knowledge and skill to solve these problems in a group.
CO2	Use standard norms of engineering practices to analyze the impact of solutions in societal and environmental context for sustainable development.
CO3	Develop interpersonal skills and ethical awareness to work as a member of a group or leader and demonstrate capabilities of self-learning in a group, which leads to lifelong learning.
CO4	Demonstrate project management principles during project work and Excel in written and oral communication.

Course Outcomes Attained

Project Title: IDENTIFICATION OF INDIAN SIGN LANGUAGE

CO No.	**COURSE OUTCOMES ATTAINED BY THE PROJECT**
CO1	Identify problems based on societal/research needs and apply knowledge and skill to solve these problems in a group.
CO2	Use standard norms of engineering practices to analyze the impact of solutions in societal and environmental context for sustainable development.
CO3	Develop interpersonal skills and ethical awareness to work as a member of a group or leader and demonstrate capabilities of self-learning in a group, which leads to lifelong learning.
CO4	Demonstrate project management principles during project work and Excel in written and oral communication.

CO-PO-PSO Mapping

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
CO1	3	3	3	3	3	3	1	3	3	3	2	3	3	3
CO2	3	3	3	3	3	3	1	3	3	3	2	3	3	3
CO3	3	3	3	3	3	3	1	3	3	3	2	3	3	3
CO4	3	3	3	3	3	3	1	3	3	3	2	3	3	3

Chapter 1

INTRODUCTION

1.1 Introduction

Sign language is a fundamental means of communication for people who are deaf or hard-of-hearing. However, many individuals in society are not proficient in sign language, leading to a communication gap between deaf individuals and the rest of society. Traditional sign language is an established method of communication, but recent technological advancements have led to the development of new techniques for identifying and generating sign language animations.

In this project, we present a novel approach for recognizing Indian sign language (ISL) alphabets using deep learning techniques and generating sign language animations from text input. The proposed method involves the use of 3D animation and natural language processing techniques to convert text input into sign language animations.

We aim to bridge the communication gap between the deaf and hearing communities by creating a system that recognizes and generates ISL animations in real-time. The project also focuses on exploring the challenges associated with ISL recognition and animation generation, including variations in hand gestures and sign language alphabets. Our approach provides a potential solution to improve communication and accessibility for deaf individuals, leading to a more inclusive society.

1.2 Problem Definition

Sign Languages are the native language and primary mode of communication. So normal people have difficulty in understanding their hand signs. Hearing or speech-disabled people who know sign language require a translator who also knows sign language to effectively communicate their thoughts to others. In order to communicate information to normal people, systems that can identify various signs are required. This is a complex problem due to the variability and complexity of sign language, which varies from region to region and even between individual signers. The goal of such a system is to provide equal access to information and communication for individuals who use sign language as their primary mode of communication.

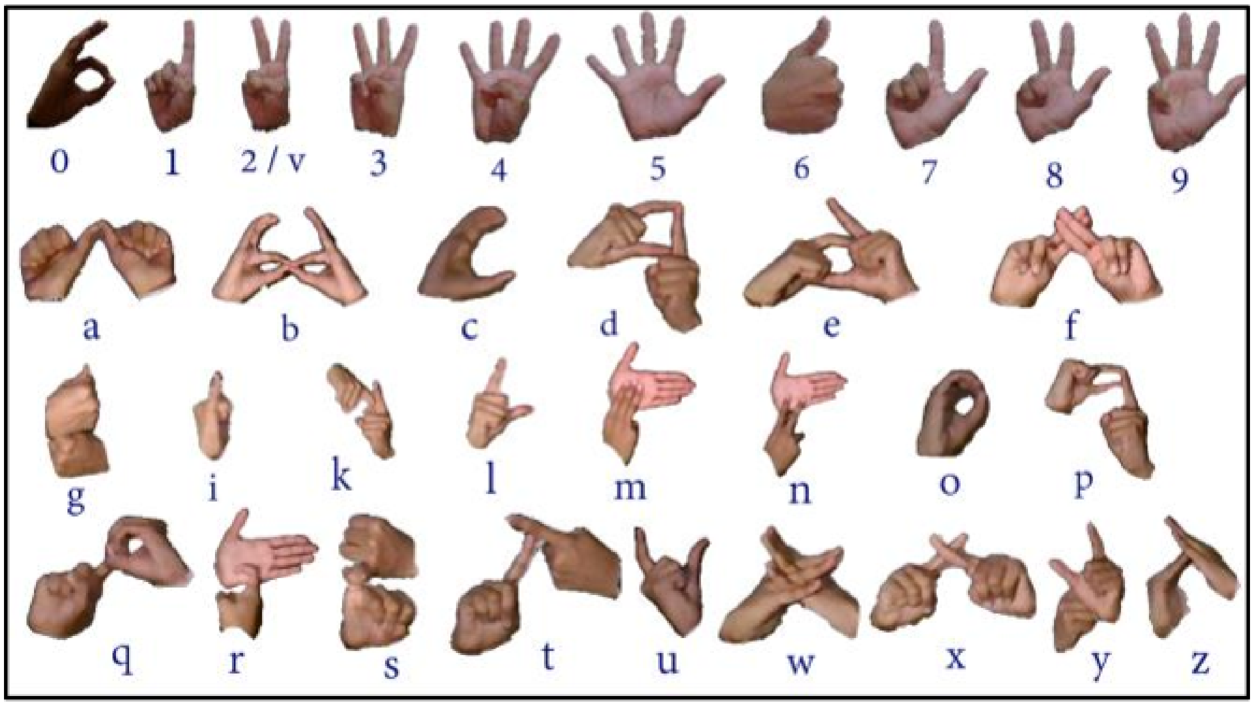


Figure 1.1: Indian Sign Language Alphabets [1].

In the Figure 1.1 hand sign shows the Indian Sign Language (ISL), which includes signs for the numbers 0 to 9 and the alphabet. ISL is a two-handed sign language, in contrast to other sign languages like American Sign Language (ASL) which use one hand only. It makes it challenging to learn.

1.3 Aim

1. The aim of identifying sign language is to create technologies capable of recognizing and interpreting sign languages used by individuals who are deaf or hard of hearing.

2. The purpose of this is to enhance communication between the hearing community and those who are deaf or hard of hearing, as well as to provide easier access to information and services.

3. The development of such systems can improve the overall quality of life for those who rely on sign language to communicate.

4. Therefore, the identification of sign language is a crucial aspect of creating an inclusive society that accommodates individuals with diverse communication needs.

1.4 Objectives

The main objectives for the Identification OF Indian Sign Language include:

1. Accurate recognition of sign language gestures: The primary objective is to develop a system that can accurately recognize and translate sign language gestures into written or spoken language. The system should be able to recognize a wide range of signs, including variations in speed, style, and hand shape.

2. Robustness to variations in signing: The system should be able to recognize sign language gestures even when they are performed by different individuals.

3. Real-time recognition: The system should be able to recognize sign language gestures in real-time, enabling users to communicate effectively and efficiently.

4. User-friendly interface: The system should have an easy-to-use interface that allows users to input sign language gestures and receive accurate translations in real-time. The system should also provide feedback to users, enabling them to improve their signing skills over time.

5. Accessibility: The system should be accessible to individuals who are communicating with deaf or hard of hearing.

The objectives of the identification of sign language are to provide equal access to information and communication for individuals who use sign language as their primary mode of communication and to improve the quality of life for these individuals.

1.5 Scope of the project

The scope of a project for identifying sign language would depend on the specific objectives and goals of the project. Here are some general aspects to consider:

1. Data collection: The project would require collecting a large dataset of sign language videos, preferably recorded from multiple sources, in different environments, with different signers and accents.

2. Annotation: The videos would need to be annotated with labels for each sign, either manually or through automated techniques such as computer vision and machine learning.

3. Sign language recognition: The project would involve developing algorithms and models for recognizing signs from the annotated videos. This could involve machine learning techniques such as deep learning, as well as rule-based approaches.

4. Evaluation: The project would need to be evaluated on a dataset of unseen sign language videos to measure the accuracy and effectiveness of the developed algorithms and models.

1.6 Existing system

Existing sign language recognition systems are primarily designed for American Sign Language (ASL), with very few focused on the Indian Sign Language (ISL). This is largely due to the fact that ISL has several variations for a particular sign, making the development of an ISL recognition system challenging. Moreover, the available ISL datasets lack rich features when compared to those available for ASL. For instance, ASL datasets have employed various approaches, such as the use of kinetic gloves or sensors worn on the body, to collect more comprehensive datasets.

Most of the current ISL recognition systems rely on vision-based recognition techniques and require the signer to perform signs within a specific region of the video frame. Furthermore, there is a lack of specific implementation for generating sign images based on given text input. Overall, the development of a robust ISL recognition system presents unique challenges that require a comprehensive approach to address the variations and complexities of the language.

Chapter 2

REVIEW OF LITERATURE

2.1 Literature Survey

Table 2.1: Summary of CNN Based Approaches Applied for Sign Language Recognition.

Year	Authors	Title	Algorithm Architecture	Performance Results
2015	Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, Benjamin Schrauwen [2]	Sign Language Recognition using Convolutional Neural Networks.	CNN, ANN	The accuracy on the test set is 95.68% and observe a 4.13% false positive rate, caused by the noise movements
2018	G.Anantha Rao K.Syamala, P.V.V.Kishore, A.S.C.S.Sastry [3].	Deep Convolutional Neural Networks for Sign Language Recognition.	CNN	Deep Convolutional Neural Networks for Sign Language Recognition. CNN Comparing the proposed CNN model to existing state-of-the-art classifiers, its average recognition rate was higher at 92.88%.
Apr 2018	Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, Weiping Li [4].	Video-Based Sign Language Recognition without Temporal Segmentation.	Two-stream 3D CNN	Proposed method LS-HAN achieves 82.7 % accuracy which is more than LSTM-E 76.8%.
June 2018	M.A Hossen, Arun Govindaiah, Sadia Sultana, Alauddin Bhuiyan [5].	Bengali Sign Language Recognition Using Deep Convolutional Neural Network	Pre-trained VGG16, CNN	Proposed recognition system results - validation loss of 0.3523 (categorical cross-entropy) and validation accuracy of 84.68%.
2018	Kshitij Bantupalli, Ying Xie [6]	American Sign Language Recognition using Deep Learning and Computer Vision.	CNN, RNN, Machine learning, HMM.	Instead of using Pool Layer, the system was able to attain 93% accuracy with SoftMax Layer.

Year	Authors	Title	Algorithm Architecture	Performance Results
Apr 2019	Sruthi C. J and Lijiya A [7].	Signet: A Deep Learning based Indian Sign Language Recognition System.	CNN	The proposed method gave training accuracy of 99.93% and validation accuracy of 98.64%.
Dec 2019	Lean Karlo S. Tolentino, Ronnie O. SerfaJuan, August C.Thio-ac, Maria Abigail B.Pamahoy, Joni Rose R. Fortezaz and Xavier Jet O. Garcia [8].	Sign language identification using Deep Learning.	CNN	The system obtained an average of 93.667%. The testing accuracy of 90.04% in letter recognition, 93.44% in number recognition, and 97.52% in static word identification.
Jan 2020	Ankita Wadhawan, Parteek Kumar [9].	Deep learning-based sign language recognition system for static signs.	CNN	A performance accuracy rate of 99.90%was calculated by the authors.
Mar 2020	Prof. Radha S. Shirbhate, Mr. Vedant D. Shinde, Ms. Sanam A. Metkari, Ms. Pooja U. Borkar, Ms. Mayuri A. Khandge [10]	Sign language Recognition Using Machine Learning Algorithm.	SVM	Initial model accuracy was 95%. One-handed and two-handed alphabet training resulted in 56% and 60% accuracy, respectively. The combined system achieved 100% accuracy on these experiments. This guarantees 100% recognition rate for previous dataset contributors.

Year	Authors	Title	Algorithm Architecture	Performance Results
2020	Dongxu Li , Chenchen Xu, Xin Yu, Kaihao Zhang, Ben Swift , Hanna Suominen, Hongdong Li [11]	TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation	TSPNet, NMT	Conv2d-RNN and TSPNet-Single outperform the state-of-the-art SLT model, Conv2d-RNN, by a large margin. The relative improvement in BLEU-4 score is 39.80% (9.58 $\rightarrow$ 13.41), and the improvement in ROUGE-L is 9.94% (31.80 $\rightarrow$ 34.96).
2020	Necati Cihan Camg,Oscar Koller, SimonHadfifield and Richard Bowden [12].	Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation.	RNN-based attention architectures, Connectionist Temporal Classification (CTC).	Compared to earlier approaches, the authors Language Transformers outperform both their recognition and their translation effectiveness with a 2% reduction in word error rate.
2021	Rachana Patil, Vivek Patil, Abhishek Bahuguna, and Mr. Gaurav Datkhile [13]	Indian Sign Language Recognition using Convolutional Neural Network.	CNN	The model achieved a validation accuracy of approximately 95%.
13 Apr 2022	Razieh Rastgoo, Kourosh Kiani, Sergio Escalera [14].	Word separation in Continuous sign language using isolated signs and post-processing	CNN, LSTM	The proposed model obtains an average of recognized Softmax outputs equal to 0.98 and 0.59.
30 Apr 2022	Abdul Mannan, Ahmed Abbasi, Abdul Rehman Javed, Anam Ahsan, Thippa Reddy Gadekallu, Qin Xin [15].	Hypertuned Deep Convolutional Neural Network for Sign Language Recognition	CNN	The proposed Deep CNN model can recognize the ASL alphabets with an accuracy rate of 99.67% on unseen test data.

Year	**Authors**	**Title**	**Algorithm Architecture**	**Performance Results**
May 2022	Suyog Bhamare, Abhishek Shinde, Rahul Palve, A. N. Gharu [16]	Sign Language Detection with CNN.	CNN	This method provides 95.7% accuracy for the 26 letters of the alphabet
June 2022	Deep Kothadiya 1, Chintan Bhatt 1, Krenil Sapariya , Kevin Patel, Ana-Belén Gil-González and Juan M. Corchado [17]	Deepsign: Sign Language Detection and Recognition Using Deep Learning	CNN, RNN, DeepL (LSTM)	This paper achieves an accuracy of 97% in sign identification.

Oct 2022	Md. Nafis Saiful, Abdulla Al Isam, Hamim Ahmed Moon, Rifa Tammana Jaman, Mitul Das, Md. Raisul Alam, Ashifur Rahman [18]	Real-Time Sign Language Detection Using CNN.	CNN	Accuracy = 98.6%
				Precision = 99%
				Recall = 99%
				F1-Score = 99%

2022	Ahmed Sultan, Walied Makram, Mohammed Kayed, Abdelmaged Amin Ali [19]	Sign language identification and recognition: A comparative study.	ML, Classical Algo., KNN.	The proposed model accuracy exceeds other classifiers such as KNN (95.95%), SVM (97.9%), and ANN (98%).

Year	Authors	Title	Algorithm Architecture	Performance Results
2022	Sharvani Srivastava, Amisha Gangwar, Richa Mishra, Sudhakar Singh [20]	Sign Language Recognition System using TensorFlow Object Detection API.	ML (Tensorflow)	The state-of-the-art method of the Indian Sign Language Recognition system achieved 93-96% accuracy. In spite of the dataset being small, the system has achieved an average confidence rate of 85.45%.
2022	Prem Selvaraj, Gokul NC, Pratyush Kumar, Mitesh Khapra [21]	OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages	CNN- RNN	This paper achieves an accuracy of 95.68% in sign identification.

(a) Sign Language Recognition using Convolutional Neural Networks [2]

According to paper [2]. To build a sign language recognition system using CNNs, a dataset of sign language gestures is collected and preprocessed, including resizing, normalization, and data augmentation. The preprocessed data is then split into training and testing sets, and a CNN model is trained using the training data.The CNN architecture typically involves several convolutional layers, followed by pooling layers and fully connected layers. The convolutional layers extract features from the input data, and the fully connected layers classify the extracted features into different sign language gestures.The trained CNN model can be used to recognize sign language gestures in real-time. The input can be captured through a camera or other sensors, and the output can be displayed to the user through a graphical user interface or other means.The advantages of using CNNs for sign language recognition include their ability to learn complex patterns and relationships in image data and their robustness to variations in the input data. CNNs have shown excellent performance in recognizing sign language gestures, outperforming traditional machine learning techniques.Sign language recognition using CNNs has many potential applications, including improving accessibility for people with disabilities, facilitating communication for deaf and hard-of-hearing individuals, and aiding in the teaching of sign language. It can also be used in human-computer interaction, robotics, and virtual reality applications [2].

(b) Deep Convolutional Neural Networks for Sign Language Recognition [3]

According to paper [3], the model is constructed with input layer, four convolutional layers, five rectified linear units (ReLu), two stochastic pooling layers, one dense and one SoftMax output layer. The CNN architecture uses four convolutional layers with different window sizes followed by an activation function, and a rectified linear unit for non-linearities. The network is trained to learn the features of each sign by means of a supervised learning. The recognition accuracy is further improved by replacing ANN with deep ANN and reported an increase in recognition rate by 5%. Hence, CNN's are a suitable tool for simulating sign language recognition on mobile platforms [3].

(c) Video-Based Sign Language Recognition without Temporal Segmentation [4]

In the proposed paper [4], Hierarchical Attention Network with Latent Space (LS-HAN) is used to translate signing videos sentence-by-sentence. Each video is divided into clips with a sliding window algorithm. The alignment information needs to be reconstructed. Empirical experiments verify that the strategy out performs others, therefore it is employed in the testing phase. After encoding, the start symbol "Start" is fed to Hierarchical Attention Network (HAN) indicating the beginning of sentence prediction. During each decoding timestamp, the word with the highest probability after the soft max is chosen as the predicted word, with its representation in the latent space fed to (HAN) for the next timestamp, until the emission of the end symbol "End" [4] .

(d) Bengali Sign Language Recognition Using Deep Convolutional Neural Network [5]

The author [5] approach was designed to function with one-handed gestures. Convolution, Max-Pooling, ReLU, Dropout, Fully Connected, and SoftMax layers form up the deep learning network. The neural network employs a stack of layers to perform classification using the features that are collected from the convolution layers. The pre-trained network is run once on the dataset to extract the features required for classification. The validation accuracy of the authors' proposed model was 84.68%, with a validation loss of 0.3523 [5].

(e) American Sign Language Recognition using Deep Learning and Computer Vision [6]

Kshitij Bantupalli, Ying Xie [6] mentioned in the paper about CNN model extracted temporal features from the frames which was used further to predict gestures based on sequence of frames. Used two different approaches to classification: a. Using the predictions from the Softmax layer and b. Using the output of the global pool layer. The global pool layer gives a 2048 sized vector, which possibly allows for more features to be analyzed by the RNN. The feature sequence is then fed to a Long-Short Term Memory (LSTM) allowing longer time dependencies. RNN's suffer from the vanishing/ exploding gradient problem, LSTM's deal with the problem allowing for higher accuracies on longer sequences of data. The model reported an incredibly high 99% accuracy on the training set. The gesture segments identified and processed by the CNN are classified by the LSTM into one of the gesture classes using sequence data. The system was able to achieve 93% accuracy with Softmax Layer rather than Pool Layer [6].

(f) Signet: A Deep Learning based Indian Sign Language Recognition System [7]

The paper [7] presented a vision-based deep learning architecture for signer independent Indian sign language recognition system. The system was successfully trained on all 24 ISL static alphabets with a training accuracy of 99.93% and with testing and validation accuracy of 98.64%. The recognition accuracy obtained is better than most of the current state of art methods [7].

(g) Sign language identification using Deep Learning [8]

The author of the paper [8] proposed a model where the network uses a Stochastic gradient descent optimizer as its optimizer to train the network having a learning rate of 1×102. The total number of epochs used to train the network is 50epochs with a batch size of 500. The images were resized to (50, 50, 1) for training and testing. To attain specific results, Keras and CNN architecture containing a set of different layers for processing of training of data used. The number of filters in the CNN layers is increased to 64. The fully connected layer is being specified by the dense layer along with rectified linear activation [8].

(h) Deep learning-based sign language recognition system for static signs [9]

Ankita Wadhawan, Parteek Kumar [9] proposed sign language recognition system includes four major phases that are data acquisition, image preprocessing, training and testing of the CNN classifier. The model training is based on convolutional neural networks. The classifier takes the preprocessed sign images and classifies it into the corresponding category. The classifier is trained on the dataset of different ISL signs. The system achieved training and validation accuracy of 99.76% and 98.35%, respectively, using RMSProp and it has been found that the SGDoptimizer outperformed Adam, RMSProp and other optimizers with training and validation accuracy of 99.90% and 98.70%, respectively, on grayscale image dataset [9].

(i) Sign language Recognition Using Machine Learning Algorithm [10]

According to paper [10] To create a Sign Language Recognition (SLR) System using machine learning, a dataset of sign language gestures is collected and augmented. The data is preprocessed, split into training and testing sets, and a machine learning model is trained. Neural networks are effective in capturing complex patterns and are commonly used in SLR systems. Once trained, the model can recognize sign language gestures in real-time through a camera or sensors. The system can improve accessibility, aid in teaching sign language, and be used in human-computer interaction, robotics, and virtual reality applications [10].

(j) TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation [11]

The paper [11] recommends using video segments rather than individual frames to train a Temporal Semantic Pyramid Network (TSPNet). The goal is to learn sign video representations that incorporate both temporal dynamics and spatial appearance. However, extracting precise gesture segments from a continuous sign video can be difficult, and noisy segments can negatively impact feature learning. The authors identify two key elements that affect the semantics of sign segments. To improve sign representations, their proposed TSPNet takes advantage of the shared semantic consistency of the segments. Specifically, the TSPNet maintains local semantic consistency by aggregating features of segments in each semantic neighborhood, using an inter-scale attention mechanism after organizing numerous video segments of various granularities in a hierarchy [11] .

(k) Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation [12]

In the paper [12], a modified version of JoeyNMT was used to implement the Sign Language Transformers. The network's components were built using PyTorch, with Transformers consisting of 512 hidden units and 8 heads in each layer. The Adam optimizer was used to train the networks with a batch size of 32, and the network was evaluated every 100 iterations [12] .

(l) Indian Sign Language Recognition using Convolutional Neural Network [13].

According to paper [13], the primary objective is to detect Indian Sign Language alphabets using their associated gestures. While gesture and sign language identification has been extensively researched in American Sign Language, it has received minimal attention in Indian Sign Language. To address this issue, the paper proposes distinguishing gestures from images, which can be obtained from a camera, instead of using high-end technologies like gloves or Kinect. Computer vision and machine learning techniques are then used to extract specific elements and classify them [13].

(m) Word separation in continuous sign language using isolated signs and post-processing [14]

The paper discusses the challenges of identifying sign boundaries in a video of continuous signs and proposes a two-step solution using transfer learning. The authors utilized a hand-crafted SVD to extract features and feed them into an LSTM network for prediction. They applied a post-processing algorithm to the output of the predictor model from the Softmax. A sliding window of 50 frames was used for both the SVD feature extractor and the hand pose estimator. The many-to-one LSTM network maps the matching SVD feature sequence into a single vector up to 50 frames, which is then passed to a fully connected layer and a Softmax layer for classification. A threshold of 0.51 was used to accept or reject a recognized class for the separation of isolated signs in a continuous sign video. The paper highlights some difficulties in recognizing certain signs such as "Congratulation," "Excuse," "Upset," "Blame," "Fight," and "Competition." [14].

(n) Hypertuned Deep Convolutional Neural Network for Sign Language Recognition [15]

In paper [15], Paper suggested an architecture based on CNN, containing dense, max-pooling, dropout, and many convolutional layers (fully connected). Which performs convolution on input with various filter and kernel sizes to map feature. Model correctly learns features from three main blocks, each of which has a different parameter configuration and uses ReLU as an activation function. By employing the flatten layer, which turns data into a vector before connecting to a group of the fully connected layer, these features were made flat. Authors evaluated the model on new data in a later stage and reported accuracy of 99.67% [15]

(o) Sign Language Detection with CNN [16]

In paper [16], to create a sign language detection system using CNNs, a dataset of sign language gestures is collected and preprocessed through resizing, normalization, and data augmentation. The preprocessed data is divided into training and testing sets, and a CNN model is trained using the training data. The CNN architecture consists of convolutional layers, pooling layers, and fully connected layers, with the convolutional layers extracting features and fully connected layers classifying them. The CNN model can detect sign language gestures in real-time using inputs captured from a camera or sensors, with CNNs having advantages in learning complex patterns and being robust to input variations. CNNs outperform traditional machine learning techniques in recognizing sign language gestures. [16]

(p) Deepsign: Sign Language Detection and Recognition Using Deep Learning [17]

According to paper [17], the authors proposed a methodology for Indian Sign Language (ISL) recognition that combines various deep learning algorithms, particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), and does not require any specific environment or camera set-up for inference. The experiments were conducted considering the real-time Indian scenario and using isolated signs, with four different combinations of LSTM and GRU used in the simulation. The suggested model performed better on frequently used terms such as "hello," "good morning," "work," etc., than any other ISL model currently in use. Furthermore, adding more layers to the LSTM and GRU and applying LSTM before GRU helped the model recognize ISL with greater accuracy [17] .

(q) Real-Time Sign Language Detection Using CNN [18]

According to paper [18], the authors processed and trained a CNN model using Google Collabs and proposed a new technique for identifying sign language. The system was divided into four steps, including real-time prediction, model training, model construction, and data preprocessing. The CNN model was chosen for its high accuracy in classification tasks and specifically utilized the Conv2D layer to detect sign language [18] .

(r) Sign language identification and recognition: A comparative study [19]

According to paper [19], the study involves collecting and preprocessing a dataset of sign language gestures, using different feature extraction techniques to represent the gestures. Various classification algorithms, including support vector machines, decision trees, neural networks, and hidden Markov models, are applied to identify and recognize the gestures. The techniques are evaluated for accuracy, speed, and robustness to variations in input data, while considering the computational complexity and practicality of the approach. The study's results can guide the development of effective sign language identification and recognition systems that improve accessibility, aid in teaching sign language, and facilitate communication for deaf and hard-of-hearing individuals [19] .

(s) Sign Language Recognition System using TensorFlow Object Detection API [20]

According to paper [20], the Sign Language Recognition System using TensorFlow and Object Detection API involves collecting a dataset of sign language gestures from various sources, preprocessing it, and building a deep learning model using TensorFlow. The trained model is optimized using techniques such as backpropagation and gradient descent. The Object Detection API is then used to recognize sign language gestures in real-time by detecting objects within the image and classifying them as sign language gestures using the SLR model. The output is displayed through a graphical user interface or other means. The system has potential applications in facilitating communication for deaf and hard-of-hearing individuals, improving accessibility for people with disabilities [20].

(t) OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages [21]

In a research paper titled "The Survey on Sign Language Recognition in the Context of Vision-Based and Deep Learning," the authors provide an overview of the current techniques and methods used for sign language recognition. The paper focuses on vision-based and deep learning approaches, which are the most popular methods for sign language recognition. It discusses various aspects of sign language recognition, such as data collection and pre-processing, feature extraction, and classification techniques. The paper also reviews different types of sign language recognition systems and their respective advantages and disadvantages. The researchers identified several challenges in sign language recognition, including the high variability in signing style and the need for large amounts of annotated data to train deep learning models. Additionally, they discussed the potential for using multimodal approaches, such as combining video and audio data, to improve the accuracy of sign language recognition [21].

Chapter 3

PROPOSED SYSTEM

3.1 Description of Dataset

To develop a vision-based approach for identifying sign language, the process involves collecting a suitable dataset, preprocessing the data to prepare it for analysis, designing the neural network model architecture, training the model on the prepared dataset, and evaluating its performance on a separate test dataset.

- Dataset 1: The first dataset used for sign language recognition tasks collected from kaggle [22] consists of 35 classes, including 26 letters and number symbols from 0-9. Out of which only letters were utilized. There are 1200 colored sample photos available for each class, in JPG format, ranging from 6.40 kb to 8kb in size. Each sample image shown in figure 3.1 is of size 128 $\times$ 128 pixels.

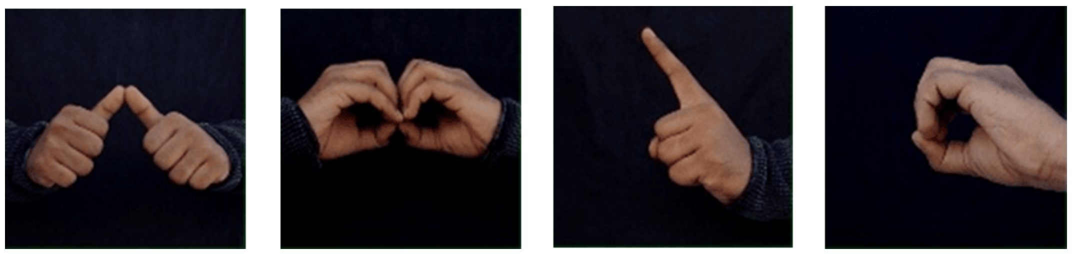


Figure 3.1: Samples from dataset 1

- Dataset 2: The second dataset used for sign language recognition contains 36,000 images, with 1000 images for each of the 36 classes. Only letter images were utilized. The images in figure 3.2 were resized to 250 x 250 pixels, with different backgrounds present in the dataset [23].

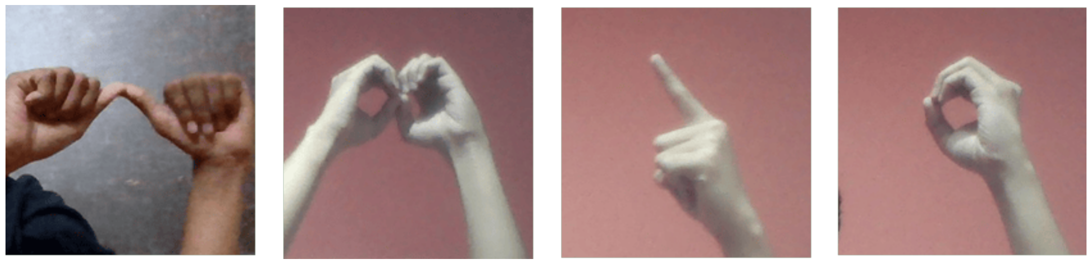


Figure 3.2: Samples from dataset 2

- Dataset 3: We generated a custom dataset of Indian Sign Language alphabet signs shown in figure 3.3 using google's mediapipe hand solutions. The dataset comprises 26 classes, each containing approximately 450 samples of 16 kb size and 250x250 resolution.

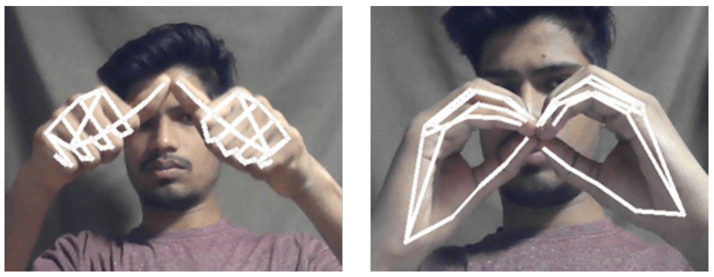


Figure 3.3: Samples from dataset 3

The datasets were divided into two separate sets, with 70% of the data allocated for training purposes and the remaining 30% reserved for testing the performance of the developed machine learning models.

3.2 Block Diagram / Architecture

Figure 3.4 shows the block diagram of Hand Sign to Text/Audio conversion. Raw hand sign images captured by cameras are preprocessed to match the specific require- ment for model processing. The CNN model is provided with features that are extracted from the output of the preprocessing phase. These features are processed by a model, which ultimately transforms and shows the recognized sign language and converts into the text/audio.

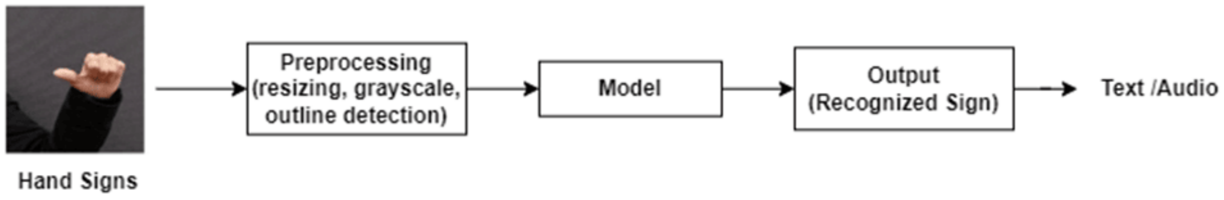


Figure 3.4: Block diagram of Hand Sign to Text/Audio conversion

Figure 3.5 shows the block diagram of Audio/Text to Hand Sign conversion. The audio input is used as raw data, which gets converted to text and sub- jected to model- based analysis. The model encodes the words into vectors provided text as input, creating the relation between words. These vectors allow for the representation of associated hand sign animations.

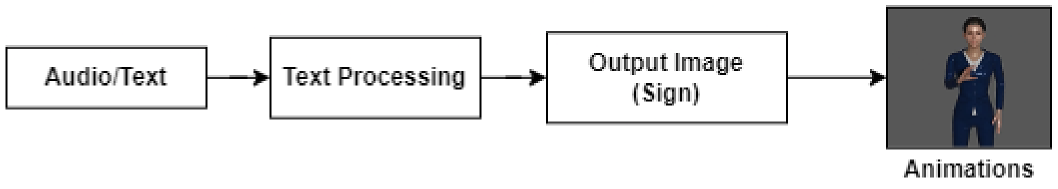


Figure 3.5: Block diagram of Text/Audio to Hand Sign Animation conversion

Chapter 4

REQUIREMENT ANALYSIS

4.1 Hardware and Software

4.1.1 Hardware

Computer/ Laptop

- minimum 4GB RAM
- CPU Core i3 proccessor 11th Generation
- Graphic Nvidia GeForce MX330.

4.1.2 Software

- VS Code
- Google Colab Platform / GPU
- GUI: HTML, CSS, Javascript

Chapter 5

MODERN TOOLS

5.1 3D Modelling Tools

5.1.1 Blender:

Blender is a free and open source 3D modelling and animation software. It supports the entire 3D pipeline, including modelling, rigging, animation, simulation, rendering, compositing, and motion tracking, as well as video editing and game development. Blender's Python scripting API is used by advanced users to customise the application and create specialised tools [24].

5.2 Python Libraries

5.2.1 TensorFlow:

TensorFlow is a flexible and efficient system for building and deploying machine learning models, which provides a variety of high-level APIs for constructing neural networks, as well as low-level primitives for implementing custom operations [25].

5.2.2 Google MediaPipe:

A framework called MediaPipe is used to create machine learning pipelines for handling time-series data, including audio, video, and other types. This multi-platform framework is compatible with embedded devices like the Raspberry Pi and Jetson Nano as well as desktop/server, Android, and iOS systems.The MediaPipe perception pipeline is called a Graph. Take the first option, Hands, as an example. We input a stream of images, and the output includes hand-rendered landmarks on the images [26].

5.2.3 NLTK:

The NLTK toolkit was created for Python users to work with NLP. We can choose from a number of text processing libraries. Using NLTK, a variety of tasks can be carried out, including tokenizing , parse tree visualization, etc [27].

5.3 Deep Learning Libraries

5.3.1 CNN:

Machine learning includes convolutional neural networks (CNNs). It is a subset of the several artificial neural network models that are employed for diverse purposes and data sets. A CNN is a particular type of network design for deep learning algorithms that is utilized for tasks like image recognition and pixel data processing [28], [29]. The structure of a CNN is comparable to the connection structure of the human brain. Similar to how the brain has billions of neurons, CNN's also have neurons, but they are structured differently as shown in Fig 5.1 A CNN performs better with image inputs and voice or audio signal inputs compared to the earlier networks.

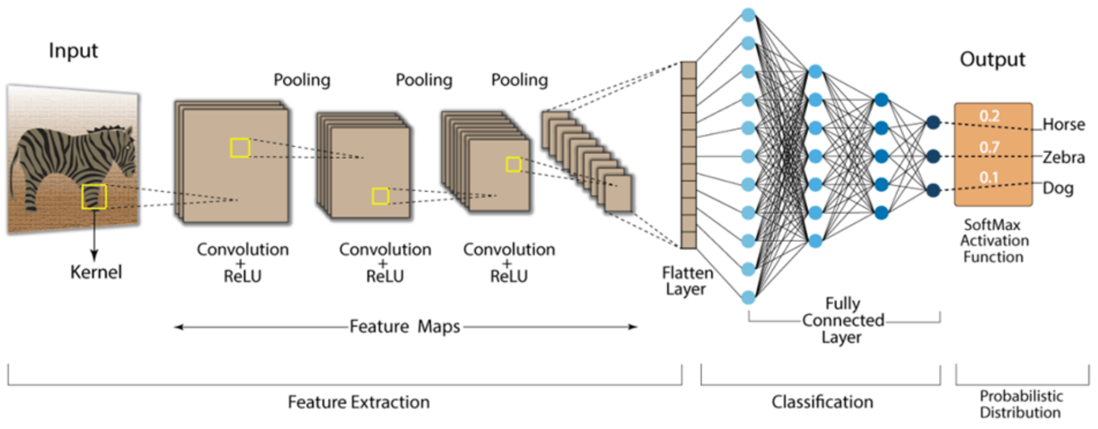


Figure 5.1: Convolution Neural Network Architecture [30]

5.3.2 KNN:

The K-Nearest Neighbors (KNN) algorithm is a simple yet effective supervised learning technique used for both regression and classification problems. It assumes that the new data is similar to the available data and classifies the new data into a category that is most similar to the available categories. KNN stores all available data and uses it to classify new data points based on their similarity to the stored data. KNN is non-parametric and does not make any assumptions about the underlying data. During the training phase, KNN stores the dataset and classifies new data based on its similarity to the stored data. Figure 5.2 illustrates the KNN Algorithm.

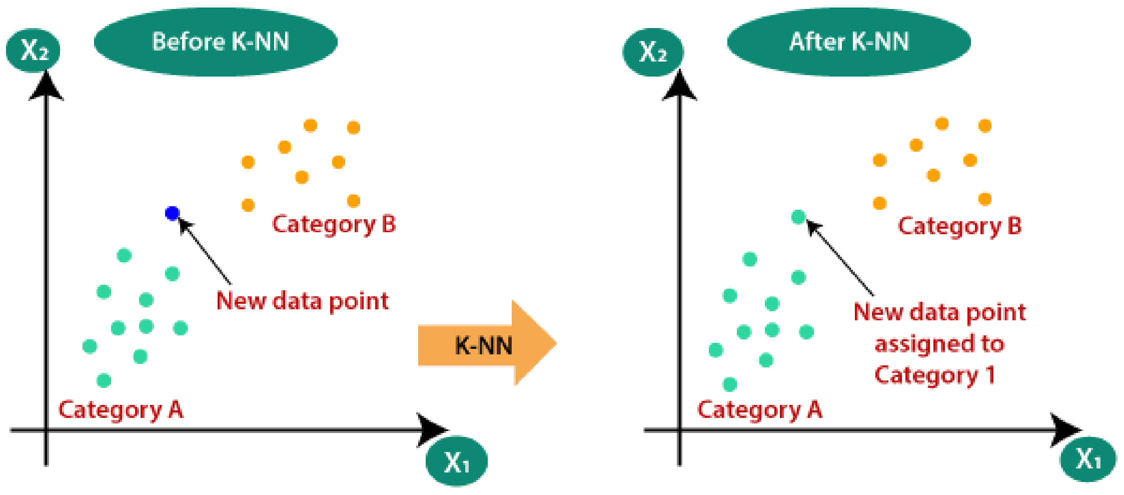


Figure 5.2: KNN Algorithm [31].

5.4 Deep Learning Optimizer

5.4.1 SGD:

Stochastic Gradient Descent (SGD) is an optimization algorithm commonly used for training deep neural networks.The term stochastic means randomness on which the algorithm is based upon. In stochastic gradient descent, instead of taking the whole dataset for each iteration, we randomly select the batches of data. That means we only take a few samples from the dataset.

$$w := w - \eta \nabla Q_i(w). \tag{5.1}$$

5.4.2 RMSprop:

The RMSprop optimizer is a variant of stochastic gradient descent (SGD) that adapts the learning rate for each parameter based on a running average of the squared gradients. Root Mean Square (RMS) is a statistical measure used to quantify the average magnitude of a set of values, typically calculated by taking the square root of the arithmetic mean of the squared values.

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{5.2}$$

5.4.3 Adam Optimizer:

Adam combines the benefits of two other optimization algorithms, namely, stochastic gradient descent (SGD) and RMSprop. It uses a combination of gradient information and a running average of the second moments of the gradients to update the parameters of the neural network.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2 \tag{5.3}$$

Chapter 6

PROPOSED METHOLOGY

6.1 Model Architecture for vision based Sign Language Identification

In this study, a Convolutional Neural Network (CNN) model that categorizes sign classes was built using a baseline common architecture. The CNN model consists of 6 convolutional layers with a filter size of 3x3 pixels and an activation function based on Rectified Linear Units (ReLU). To improve network stability, the input layer conducts input data normalization after receiving the image data. Batch normalization is then put into practice. After the first convolutional layer, average pooling is also utilized, using a (2, 2) receptive field and a 2 pixel stride. The feature maps' dimensionality is reduced by using both the average and maximum pooling operators

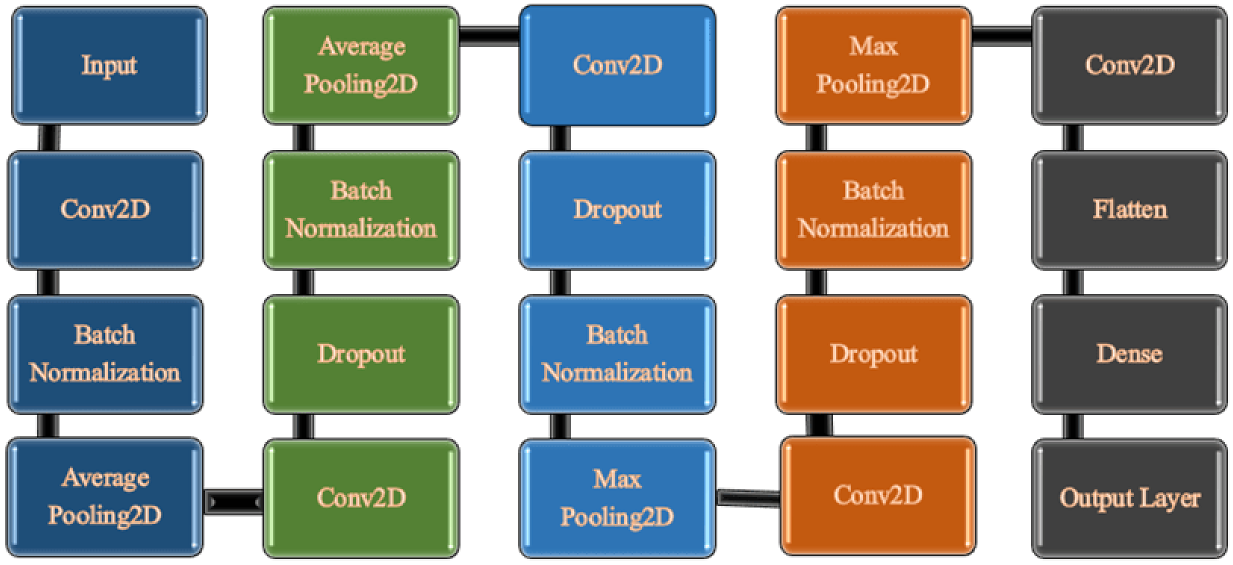


Figure 6.1: Proposed CNN Model Architecture.

In order to prevent overfitting, the second convolutional layer incorporates dropout with a rate ranging from 0.25 to 0.5, which randomly eliminates neurons during training. The feature maps are then further downsampled using average pooling. Similar patterns are used in the 3rd and 4th layers, although they employ maximum pooling rather than average pooling shown in figure 6.1. The fourth layer's output is then flattened and sent to the last dense layer, which is completely linked and computes the classification scores.

CNN Based classifier: Fundamental characteristics are taken from the preprocessed image. The classifier uses these feature vectors to identify sign classes.

Loss Function: The decision to retrain the model is made based on a threshold value. For retraining, the loss function determines the loss value. A loss function compares the target and anticipated output values. To determine how well the neural network mimics the training data.

Optimizer: An optimizer is a key component in training neural networks. It is a mathematical algorithm used to adjust the weights and biases of the model during the learning process to minimize the loss function. The optimizer performs this task by iteratively updating the model parameters to improve the accuracy of the model's predictions. The figure 6.2 illustrates the training process of the CNN based model .

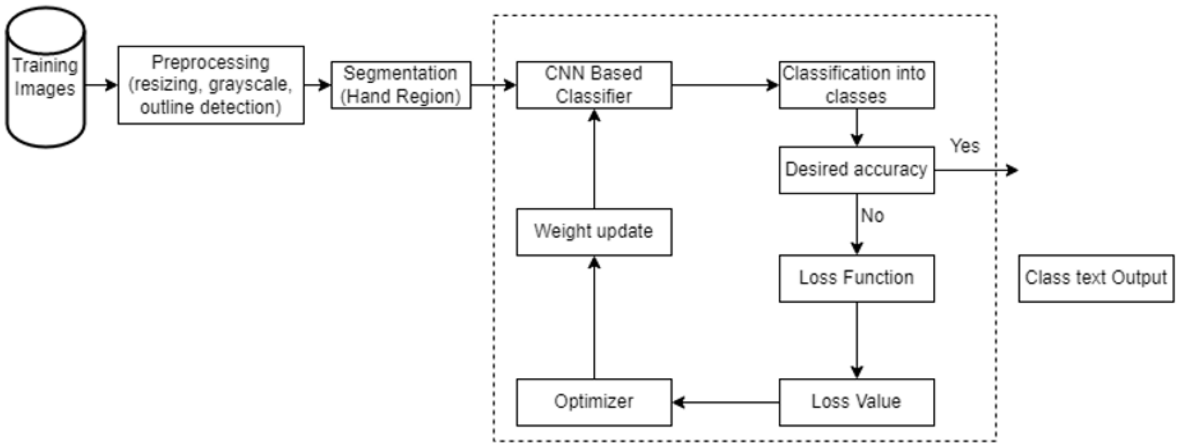


Figure 6.2: Model Training

The aim is to build a comprehensive system for sign language recognition and text to animation using Natural Language Toolkit (NLTK) on Django, which leverages the latest advancements in computer vision and natural language processing. Our method comprises multiple modules, each designed to address a specific challenge in the process, such as hand tracking, sign language recognition, and text processing on input sentences.

6.2 Text to Sign Language animation

In this module, NLTK library is utilized to preprocess the input sentence. The preprocessing includes four main steps: Tokenization, Stopword elimination, Stemming, and Lemmatization.

1. Tokenizing: It involves dividing the provided text into token-sized pieces. The tokens might represent letters, numbers, punctuation, or even complete phrases. Word boundaries are located through tokenization to accomplish this task. Word boundaries are the locations where one word ends and the next word begins. Word segmentation is another name for tokenization.

2. Stop words: They are common English expressions like ”and,” ”the,” ”a,” and ”an” that have very little significance. Basically, we remove stopwords from natural language text during preprocessing because they are repetitive and don't add any significance to the information.

3. Stemming: It is the procedure of changing infected or derivative words back to their word stem, base, or root form. Essentially, it joins prefixes, suffixes, and lemma, the base of words.In natural language processing, it serves as a preprocessing step as well.

4. Lemmatization: Is a natural language processing technique used to reduce words to their base form, or lemma. It involves identifying the base form of a word based on its context and part of speech, in order to simplify and standardize text.

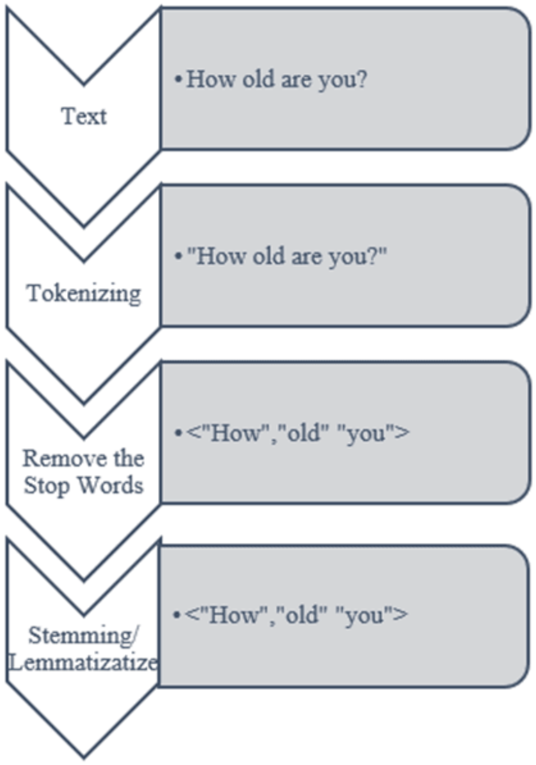


Figure 6.3: NLTK Process for Input sentence

Figure 6.3 illustrates the NLTK process used to extract essential keywords from the input sentence.

Chapter 7

IMPLEMETATION

The optimization of a Convolutional Neural Network (CNN) model requires experimentation with different hyperparameters, such as the choice of optimizers, activation functions, and the number of layers.

7.1 Selection of Optimizer

- A 6-layered CNN model was utilized for the experiment.
- Optimizers: SGD, RMSprop and Adam
- Activation function: Rectified LineraUnit (relu)

Preprocessing approach: The figure 7.1 depicts the transformations that are applied to the input images in the preprocessing step for sign language recognition.

1. Gaussian blur: It is an image smoothing technique that utilizes convolution with a Gaussian filter kernel to eliminate high-frequency components and remove noise and irrelevant details.
2. Gray scaling: It is the conversion of an image from color to grayscale, simplifying its representation and reducing its dimensionality to enhance performance and decrease computational complexity in machine learning and computer vision tasks.

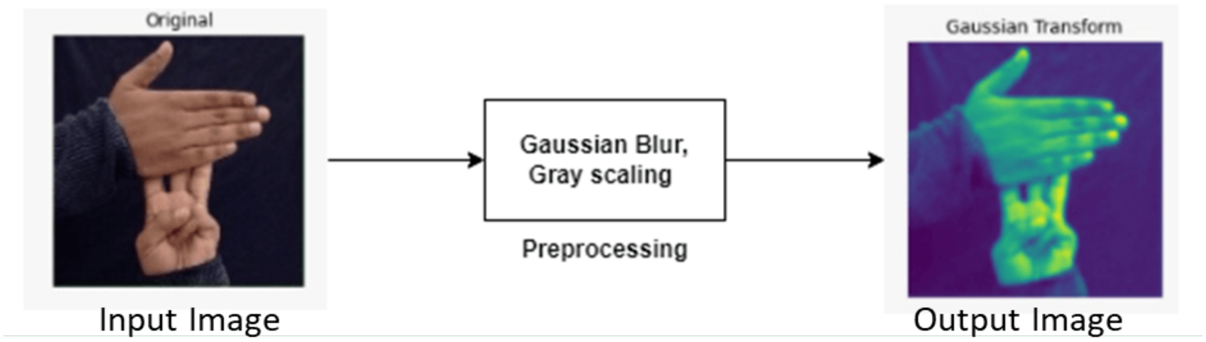


Figure 7.1: Preprocessing approach for Model 1

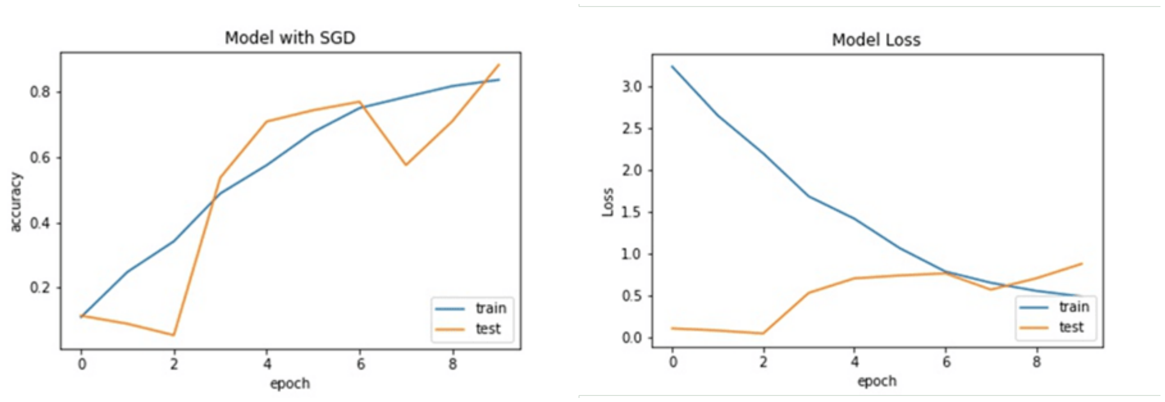


Figure 7.2: SGD Optimizer accuracy 90.0%

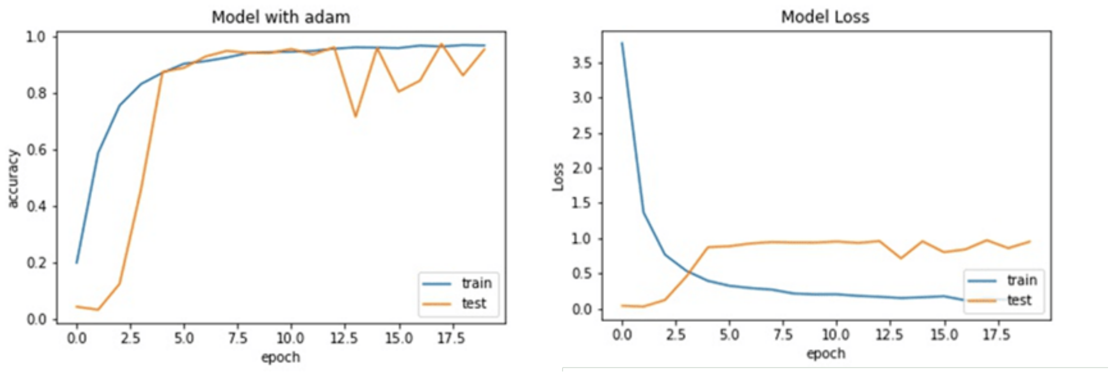


Figure 7.3: RMSprop Optimizer 92.80%

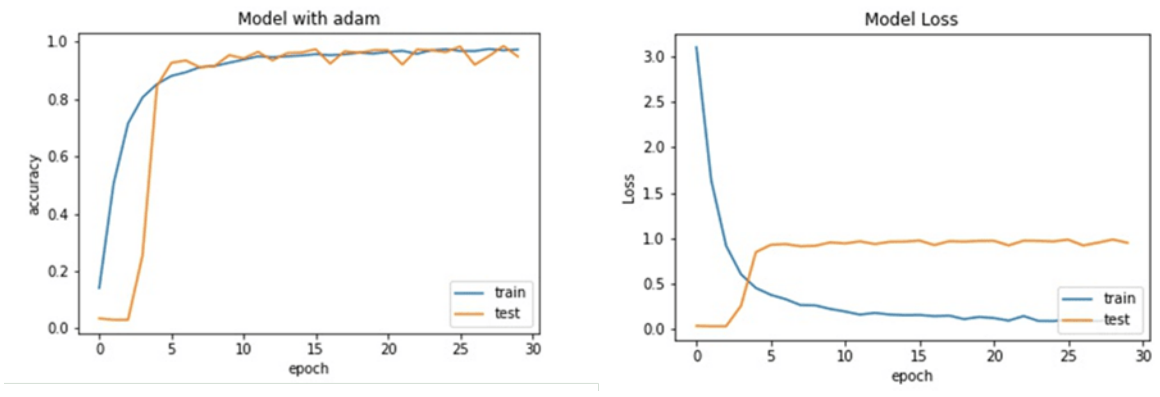


Figure 7.4: Adam Optimizer 93.50%

The training graphs, as shown in figures 7.2, 7.3, and 7.4, illustrate that the Adam optimizer outperformed the SGD and RMSprop optimizers.

Further evaluation of these models using real-time video frames revealed that Adam performed better than the other optimizers. The aim of this experiment was to select the optimal optimizer for sign language recognition. Figure 7.5 shows the result of optimizers on a real time frame.

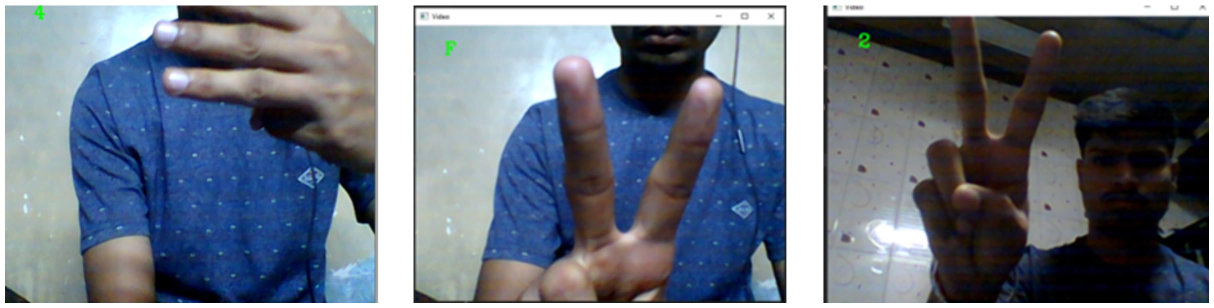


Figure 7.5: Model Result with SGD,RMSprop and Adam real time video frame

The model architecture shown in figure 6.1 was used for additional experiments with the Adam optimizer and ReLU activation function, after the initial experiments.

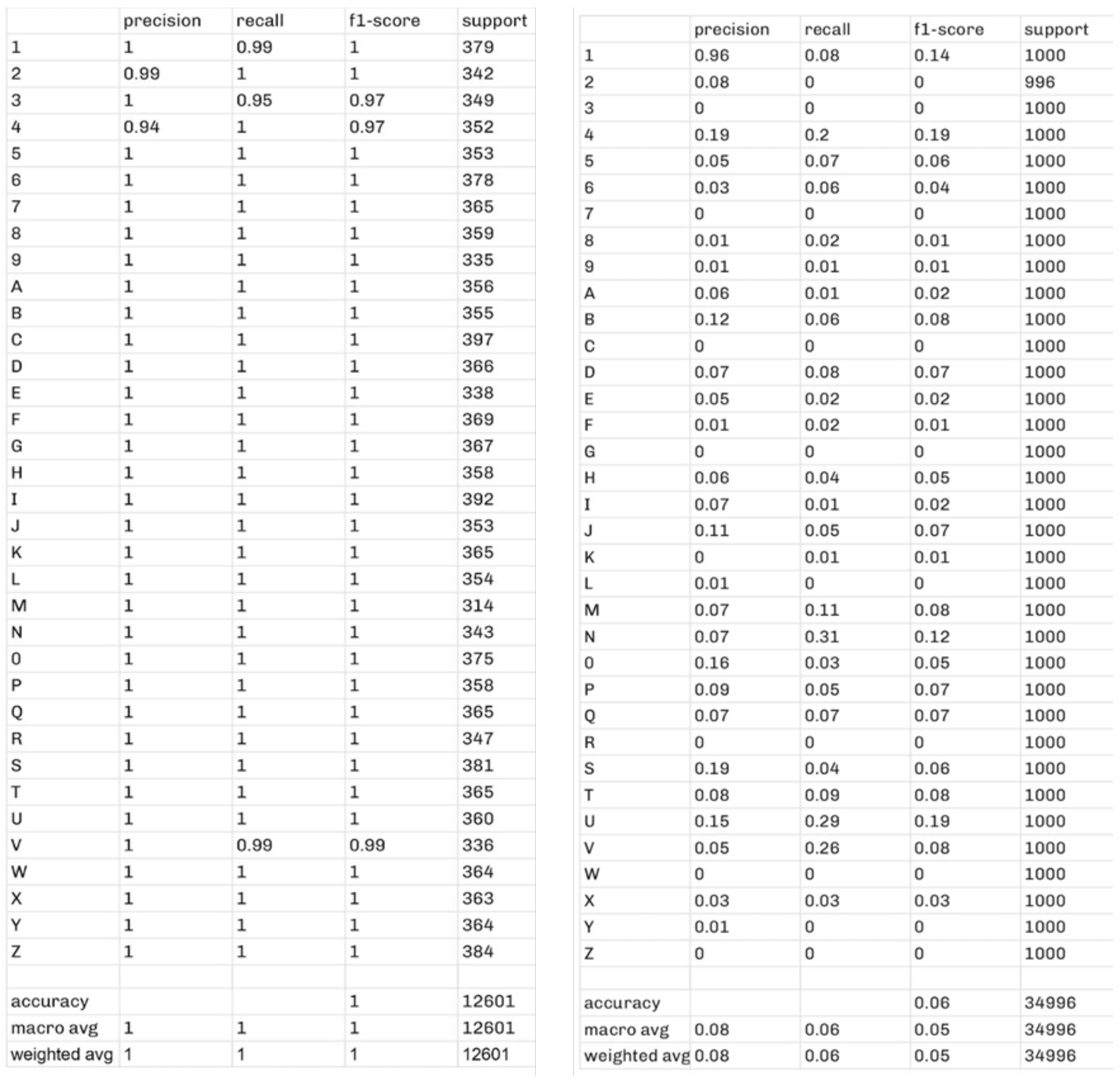
The experiment is conducted by applying various preprocessing techniques such as Gaussian blur and gray scaling, brightness and contrast level adjustments, and edge detection techniques to improve computational efficiency.

The model's performance is assessed after each modification to evaluate the efficacy of the preprocessing techniques.

7.2 Model 1

Preprocessing: The preprocessing pipeline for Model 2 remained identical to that of Model 1.

- Gaussian Blur
- Gray scaling



	precision	recall	f1-score	support
1	1	0.99	1	379
2	0.99	1	1	342
3	1	0.95	0.97	349
4	0.94	1	0.97	352
5	1	1	1	353
6	1	1	1	378
7	1	1	1	365
8	1	1	1	359
9	1	1	1	335
A	1	1	1	356
B	1	1	1	355
C	1	1	1	397
D	1	1	1	366
E	1	1	1	338
F	1	1	1	369
G	1	1	1	367
H	1	1	1	358
I	1	1	1	392
J	1	1	1	353
K	1	1	1	365
L	1	1	1	354
M	1	1	1	314
N	1	1	1	343
O	1	1	1	375
P	1	1	1	358
Q	1	1	1	365
R	1	1	1	347
S	1	1	1	381
T	1	1	1	365
U	1	1	1	360
V	1	0.99	0.99	336
W	1	1	1	364
X	1	1	1	363
Y	1	1	1	364
Z	1	1	1	384
accuracy			1	12601
macro avg	1	1	1	12601
weighted avg	1	1	1	12601

	precision	recall	f1-score	support
1	0.96	0.08	0.14	1000
2	0.08	0	0	996
3	0	0	0	1000
4	0.19	0.2	0.19	1000
5	0.05	0.07	0.06	1000
6	0.03	0.06	0.04	1000
7	0	0	0	1000
8	0.01	0.02	0.01	1000
9	0.01	0.01	0.01	1000
A	0.06	0.01	0.02	1000
B	0.12	0.06	0.08	1000
C	0	0	0	1000
D	0.07	0.08	0.07	1000
E	0.05	0.02	0.02	1000
F	0.01	0.02	0.01	1000
G	0	0	0	1000
H	0.06	0.04	0.05	1000
I	0.07	0.01	0.02	1000
J	0.11	0.05	0.07	1000
K	0	0.01	0.01	1000
L	0.01	0	0	1000
M	0.07	0.11	0.08	1000
N	0.07	0.31	0.12	1000
O	0.16	0.03	0.05	1000
P	0.09	0.05	0.07	1000
Q	0.07	0.07	0.07	1000
R	0	0	0	1000
S	0.19	0.04	0.06	1000
T	0.08	0.09	0.08	1000
U	0.15	0.29	0.19	1000
V	0.05	0.26	0.08	1000
W	0	0	0	1000
X	0.03	0.03	0.03	1000
Y	0.01	0	0	1000
Z	0	0	0	1000
accuracy			0.06	34996
macro avg	0.08	0.06	0.05	34996
weighted avg	0.08	0.06	0.05	34996

Figure 7.6: Model report on 30% Training data(Dataset 1) & Unseen data (Dataset 2)

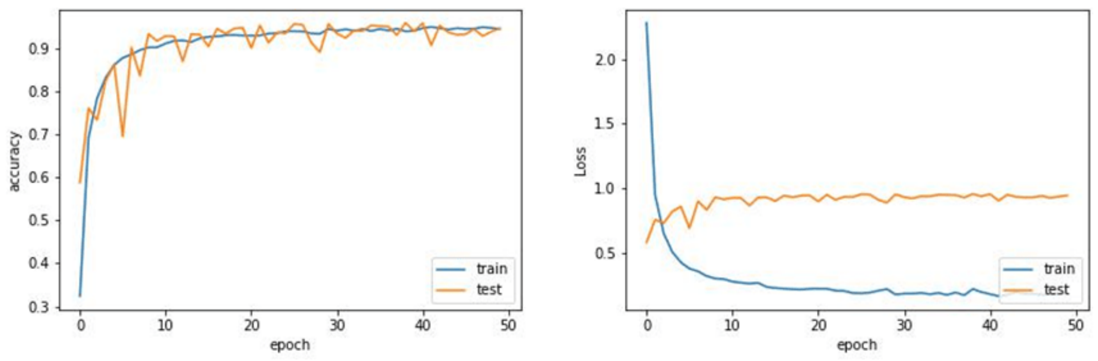


Figure 7.7: Training and Validation accuracy & Model Loss

Figures 7.6 and 7.7 demonstrate that the model's performance is unsatisfactory when evaluated on Dataset 2 and in real-time video at 30 frames per second. A meticulous investigation of the preprocessed Dataset 2 uncovered the presence of considerable noise in the images, resulting from a bright background matching the hand color shown in figure 7.8 leading to additional noise on the preprocessed images . These observations highlight the critical role of noise reduction in image preprocessing to achieve accurate recognition of hand gestures.

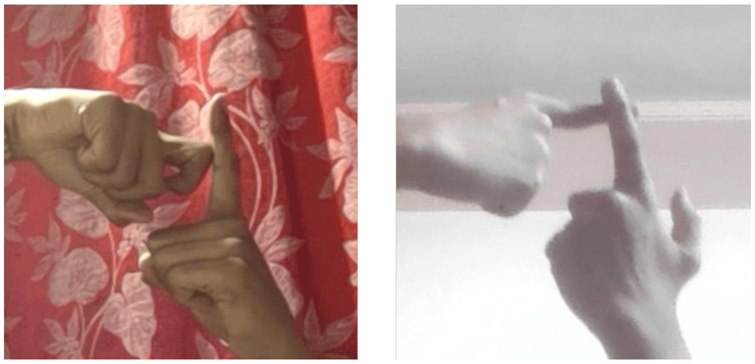


Figure 7.8: Sample Images from dataset 2

7.3 Model 2

Preprocessing:

- Gaussian Blur
- Gray scaling
- Thresholding
- Edge detection

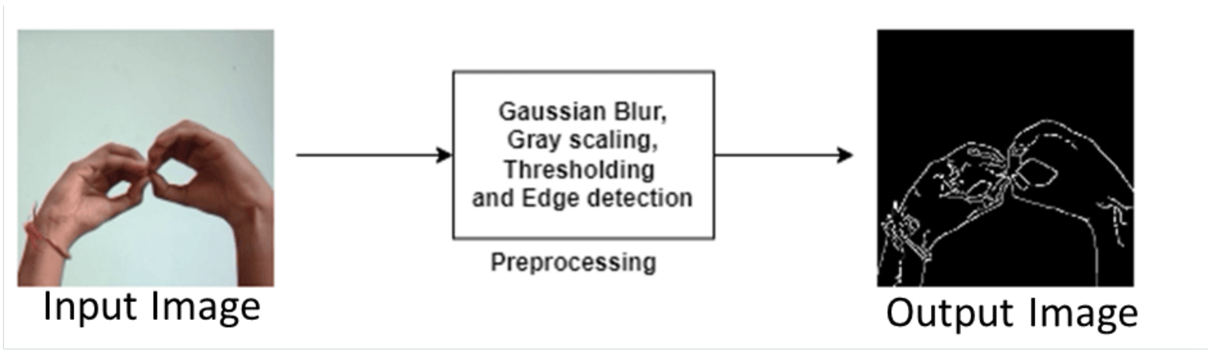
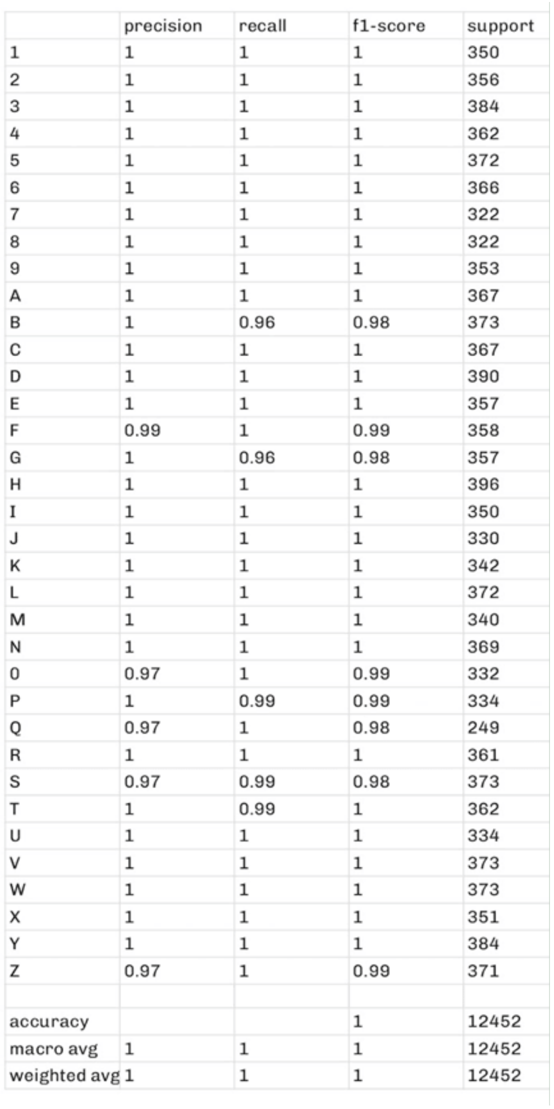
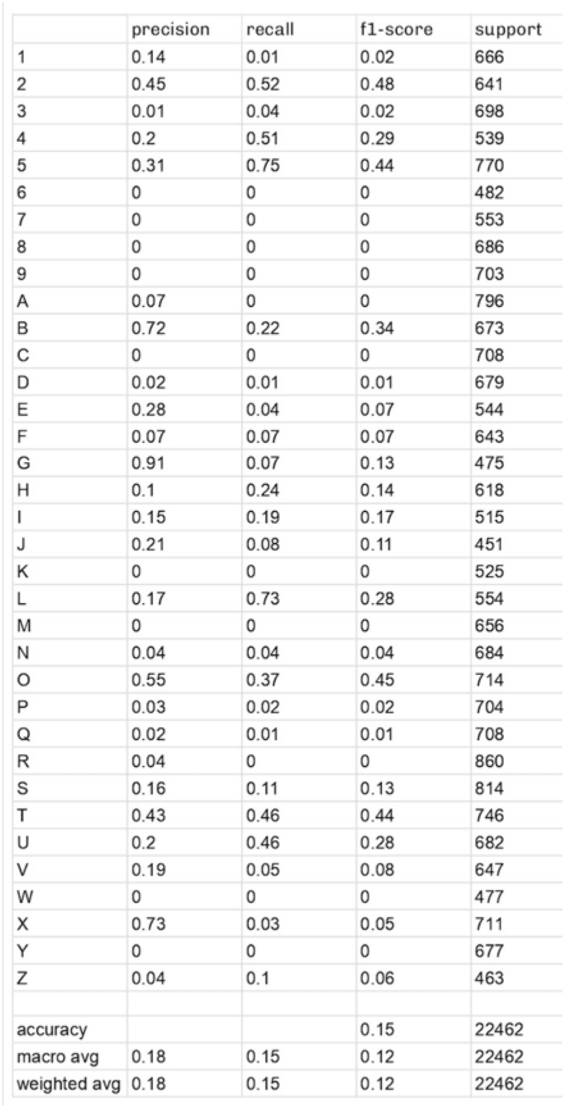


Figure 7.9: Transformed image

To tackle the challenge of hand pose variation, an augmentation process is applied to the preprocessed images using various techniques such as rotation by 10 degrees, width shift by 10%, height shift by 10%, and zoom by 10%. These augmentation techniques enhance the robustness of the model and increase its ability to generalize to various hand poses. In both Dataset 1 and 2, we adopted the preprocessing approach mentioned in above figure 7.9, which consisted of multiple image transformations. The first transformation applied was a Gaussian blur filter, which reduced the high-frequency components in the image to remove noise. Next, we applied gray-scaling to convert the color image into a grayscale image, followed by thresholding to convert the grayscale image into a binary image. Finally, we applied edge detection techniques to extract only the necessary features from the images while eliminating unnecessary data. This approach resulted in a transformed image that only contains the relevant information for hand gesture recognition.



	precision	recall	f1-score	support
1	1	1	1	350
2	1	1	1	356
3	1	1	1	384
4	1	1	1	362
5	1	1	1	372
6	1	1	1	366
7	1	1	1	322
8	1	1	1	322
9	1	1	1	353
A	1	1	1	367
B	1	0.96	0.98	373
C	1	1	1	367
D	1	1	1	390
E	1	1	1	357
F	0.99	1	0.99	358
G	1	0.96	0.98	357
H	1	1	1	396
I	1	1	1	350
J	1	1	1	330
K	1	1	1	342
L	1	1	1	372
M	1	1	1	340
N	1	1	1	369
O	0.97	1	0.99	332
P	1	0.99	0.99	334
Q	0.97	1	0.98	249
R	1	1	1	361
S	0.97	0.99	0.98	373
T	1	0.99	1	362
U	1	1	1	334
V	1	1	1	373
W	1	1	1	373
X	1	1	1	351
Y	1	1	1	384
Z	0.97	1	0.99	371
accuracy			1	12452
macro avg	1	1	1	12452
weighted avg	1	1	1	12452



	precision	recall	f1-score	support
1	0.14	0.01	0.02	666
2	0.45	0.52	0.48	641
3	0.01	0.04	0.02	698
4	0.2	0.51	0.29	539
5	0.31	0.75	0.44	770
6	0	0	0	482
7	0	0	0	553
8	0	0	0	686
9	0	0	0	703
A	0.07	0	0	796
B	0.72	0.22	0.34	673
C	0	0	0	708
D	0.02	0.01	0.01	679
E	0.28	0.04	0.07	544
F	0.07	0.07	0.07	643
G	0.91	0.07	0.13	475
H	0.1	0.24	0.14	618
I	0.15	0.19	0.17	515
J	0.21	0.08	0.11	451
K	0	0	0	525
L	0.17	0.73	0.28	554
M	0	0	0	656
N	0.04	0.04	0.04	684
O	0.55	0.37	0.45	714
P	0.03	0.02	0.02	704
Q	0.02	0.01	0.01	708
R	0.04	0	0	860
S	0.16	0.11	0.13	814
T	0.43	0.46	0.44	746
U	0.2	0.46	0.28	682
V	0.19	0.05	0.08	647
W	0	0	0	477
X	0.73	0.03	0.05	711
Y	0	0	0	677
Z	0.04	0.1	0.06	463
accuracy			0.15	22462
macro avg	0.18	0.15	0.12	22462
weighted avg	0.18	0.15	0.12	22462

Figure 7.10: Model 2 report on 30% Training data(Dataset 1) & Unseen data (Dataset 2)

The evaluation results for model 2 on unseen data from figure 7.10 revealed that it correctly classifies all labels for dataset 1, which was the dataset used for model training. However, the model's performance was unsatisfactory when tested on dataset 2, with an accuracy of only approximately 15%. Further investigation into the model's report revealed that certain classes in dataset 2 had a precision of 0, indicating that the model failed to correctly identify those signs. Upon comparison of the signs in dataset 1 and 2, it was observed that they were not identical and had different orientations.

This indicates that the model's performance is affected by the differences in sign orientation between the two datasets. Additionally, the low precision for certain classes suggests that there may be a need for further optimization of the model's training parameters or augmentation techniques to better account for variations in sign orientation shown in figure 7.11.

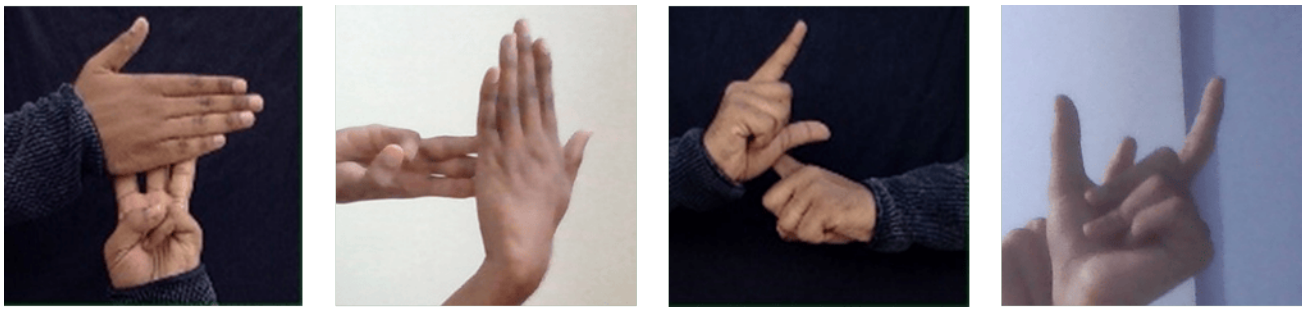


Figure 7.11: Sample images with different orientations

The results obtained indicate that the preprocessed dataset 2 has high noise due to the presence of white colors in the background pattern. The selection of an optimal threshold value for the preprocessing step posed a challenge since the same value did not work equally well for both datasets shown in figure 7.12 . Specifically, the applied threshold value removed some lower pixel values that are necessary for edge detection, leading to poor detection of edges for some classes. These findings highlight the importance of selecting an optimal threshold value for each dataset, and that a one-size-fits-all approach may not be effective.

Figure 7.12: Loss of Important features

7.4 Model 3

- Gaussian Blur
- Gray scaling
- Edge detection
- Changing Augmentation parameters

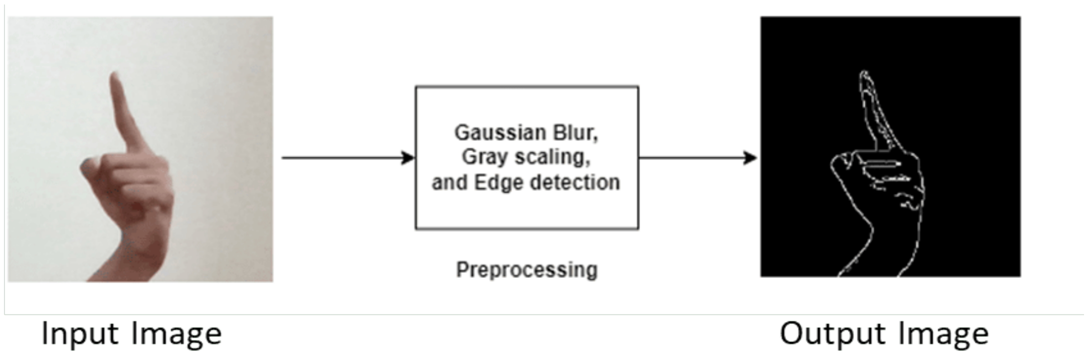
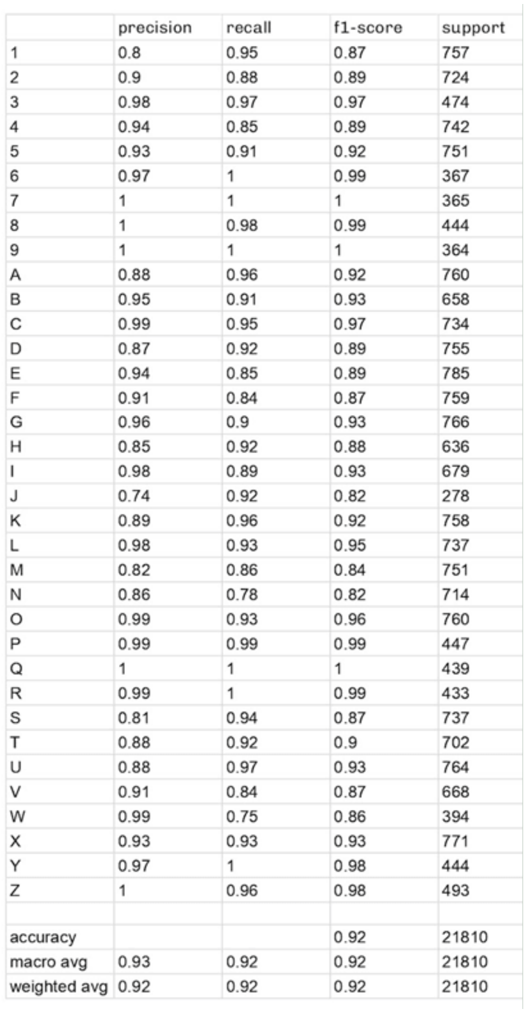


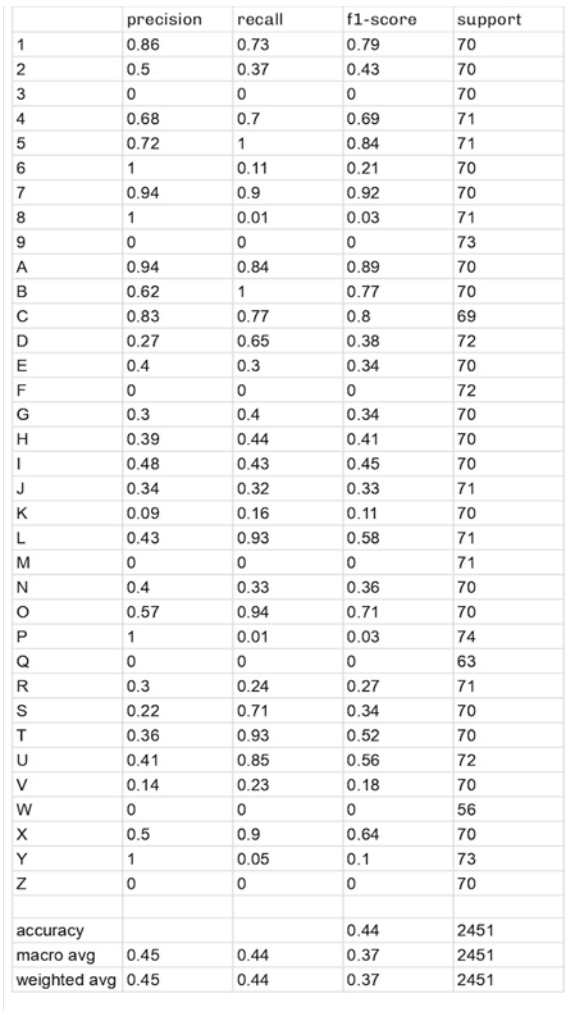
Figure 7.13: Improved edge detection

As an additional step, we incorporated 30% of dataset 2 into the training set to increase the variety of samples. This is done to prevent overfitting of the model on the training data and to improve its performance on unseen data. By diluting the training dataset with a portion of dataset 2, we aimed to capture a broader range of variations and complexities that might exist in the real-world scenario. This step helped to improve the generalizability of the model, thereby increasing its accuracy on unseen data.

The thresholding method used during the image data preprocessing stage is discovered to cause a loss of crucial features illustrated in figure 7.9. Therefore, the approach was discarded, and the effects were assessed. It was observed that this modification resulted in a better edge detection of the hand signs, while still maintaining the important features of the images shown in figure 7.13. This enhancement is significant because it can improve the performance of the model and allow for more precise detection of hand signs in real-world situations.



	precision	recall	f1-score	support
1	0.8	0.95	0.87	757
2	0.9	0.88	0.89	724
3	0.98	0.97	0.97	474
4	0.94	0.85	0.89	742
5	0.93	0.91	0.92	751
6	0.97	1	0.99	367
7	1	1	1	365
8	1	0.98	0.99	444
9	1	1	1	364
A	0.88	0.96	0.92	760
B	0.95	0.91	0.93	658
C	0.99	0.95	0.97	734
D	0.87	0.92	0.89	755
E	0.94	0.85	0.89	785
F	0.91	0.84	0.87	759
G	0.96	0.9	0.93	766
H	0.85	0.92	0.88	636
I	0.98	0.89	0.93	679
J	0.74	0.92	0.82	278
K	0.89	0.96	0.92	758
L	0.98	0.93	0.95	737
M	0.82	0.86	0.84	751
N	0.86	0.78	0.82	714
O	0.99	0.93	0.96	760
P	0.99	0.99	0.99	447
Q	1	1	1	439
R	0.99	1	0.99	433
S	0.81	0.94	0.87	737
T	0.88	0.92	0.9	702
U	0.88	0.97	0.93	764
V	0.91	0.84	0.87	668
W	0.99	0.75	0.86	394
X	0.93	0.93	0.93	771
Y	0.97	1	0.98	444
Z	1	0.96	0.98	493
accuracy			0.92	21810
macro avg	0.93	0.92	0.92	21810
weighted avg	0.92	0.92	0.92	21810



	precision	recall	f1-score	support
1	0.86	0.73	0.79	70
2	0.5	0.37	0.43	70
3	0	0	0	70
4	0.68	0.7	0.69	71
5	0.72	1	0.84	71
6	1	0.11	0.21	70
7	0.94	0.9	0.92	70
8	1	0.01	0.03	71
9	0	0	0	73
A	0.94	0.84	0.89	70
B	0.62	1	0.77	70
C	0.83	0.77	0.8	69
D	0.27	0.65	0.38	72
E	0.4	0.3	0.34	70
F	0	0	0	72
G	0.3	0.4	0.34	70
H	0.39	0.44	0.41	70
I	0.48	0.43	0.45	70
J	0.34	0.32	0.33	71
K	0.09	0.16	0.11	70
L	0.43	0.93	0.58	71
M	0	0	0	71
N	0.4	0.33	0.36	70
O	0.57	0.94	0.71	70
P	1	0.01	0.03	74
Q	0	0	0	63
R	0.3	0.24	0.27	71
S	0.22	0.71	0.34	70
T	0.36	0.93	0.52	70
U	0.41	0.85	0.56	72
V	0.14	0.23	0.18	70
W	0	0	0	56
X	0.5	0.9	0.64	70
Y	1	0.05	0.1	73
Z	0	0	0	70
accuracy			0.44	2451
macro avg	0.45	0.44	0.37	2451
weighted avg	0.45	0.44	0.37	2451

Figure 7.14: Model 3 report on 30% Training data(Dataset 1) & Unseen data (Dataset 2)

On analyzing report in figure 7.14, it is observed that incorporating an augmentation step resulted in an improvement in the accuracy of dataset 2. Despite this improvement, the model's performance on the unseen dataset is only 44%, indicating that further optimization is required. It is worth noting that adding dataset 2 to the training set increased the dataset's size and variation, leading to a decrease in accuracy to 92% when using a 30% training set. Additionally, this caused a significant increase in the memory usage of Google Colab, leading to notebook crashes during dataset loading and training.

7.5 Model 4

Preprocessing: The preprocessing pipeline for Model 4 remained identical to that of Model 3.

- Gaussian Blur
- Gray scaling
- Edge detection

For Model 4, a combined training set was created by utilizing 70% of Dataset 1 and 70% of Dataset 2. This approach is chosen to incorporate a more diverse set of hand signs into the training set, which could potentially improve the model's performance. However, due to the large size of the combined dataset, the training process required more memory than what was available in the free tier of Google Colab. Specifically, the training process required 32 GB of RAM, while the free tier was limited to 15 GB. To circumvent this issue, a decision is made to reduce the number of classes used for training by removing the classes associated with numbers 1-9 from the training set. This step allowed for the training process to be executed within the available memory resources. However, it is important to note that this approach may have affected the overall performance of the model since some of the hand sign classes are removed from the training set. The remaining 30% of both datasets were used as the unseen testing set, allowing for the evaluation of the model's performance on data that it had not seen during training.

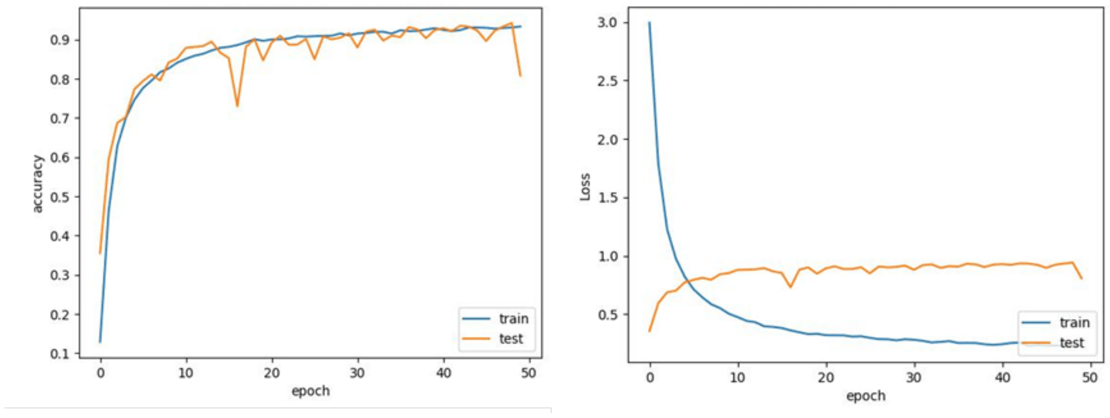
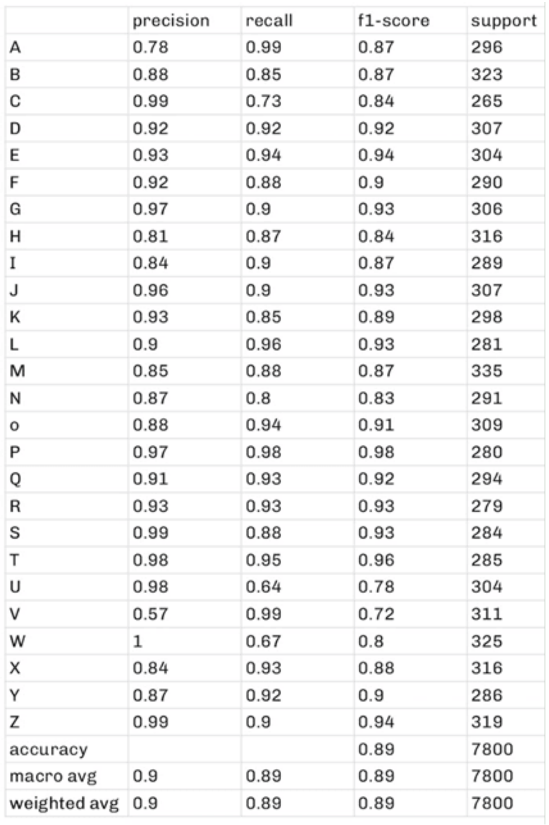
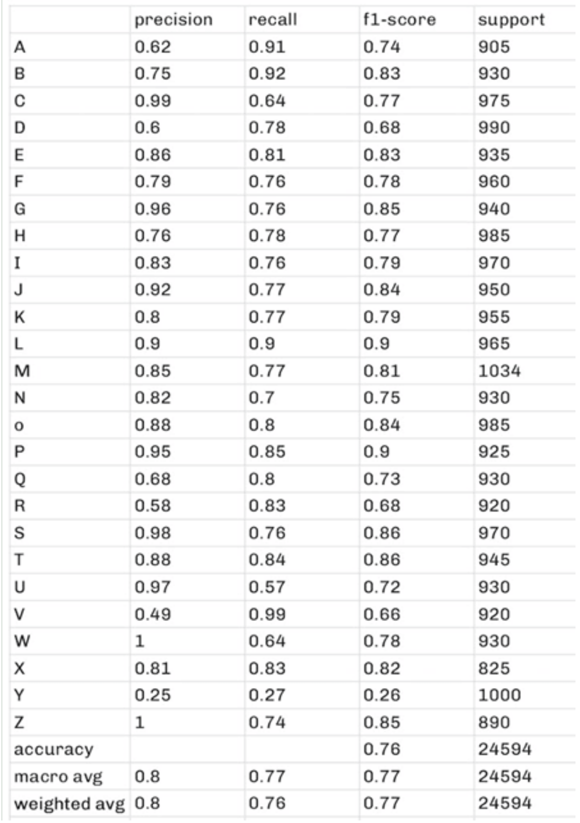


Figure 7.15: Model 4 train-test plot

In order to optimize the model parameters and number of layers for Model 4, we utilized the RandomSearch method from Keras Tuner, a hyperparameter tuning tool. During this process, we conducted 10 trials, each consisting of 10 epochs, while altering various hyperparameters such as the number of filters per layer (ranging from 32 to 256), kernel size (ranging from 2 to 5), and including dropout layers and pooling layer parameters. Following the trials, we selected the model architecture that resulted in the highest accuracy and proceeded to train it for an additional 50 epochs shown in figure 7.15. This approach allowed us to determine the optimal hyperparameters for Model 4, which can enhance its performance and accuracy in detecting hand signs in real-world scenarios.



	precision	recall	f1-score	support
A	0.78	0.99	0.87	296
B	0.88	0.85	0.87	323
C	0.99	0.73	0.84	265
D	0.92	0.92	0.92	307
E	0.93	0.94	0.94	304
F	0.92	0.88	0.9	290
G	0.97	0.9	0.93	306
H	0.81	0.87	0.84	316
I	0.84	0.9	0.87	289
J	0.96	0.9	0.93	307
K	0.93	0.85	0.89	298
L	0.9	0.96	0.93	281
M	0.85	0.88	0.87	335
N	0.87	0.8	0.83	291
o	0.88	0.94	0.91	309
P	0.97	0.98	0.98	280
Q	0.91	0.93	0.92	294
R	0.93	0.93	0.93	279
S	0.99	0.88	0.93	284
T	0.98	0.95	0.96	285
U	0.98	0.64	0.78	304
V	0.57	0.99	0.72	311
W	1	0.67	0.8	325
X	0.84	0.93	0.88	316
Y	0.87	0.92	0.9	286
Z	0.99	0.9	0.94	319
accuracy			0.89	7800
macro avg	0.9	0.89	0.89	7800
weighted avg	0.9	0.89	0.89	7800



	precision	recall	f1-score	support
A	0.62	0.91	0.74	905
B	0.75	0.92	0.83	930
C	0.99	0.64	0.77	975
D	0.6	0.78	0.68	990
E	0.86	0.81	0.83	935
F	0.79	0.76	0.78	960
G	0.96	0.76	0.85	940
H	0.76	0.78	0.77	985
I	0.83	0.76	0.79	970
J	0.92	0.77	0.84	950
K	0.8	0.77	0.79	955
L	0.9	0.9	0.9	965
M	0.85	0.77	0.81	1034
N	0.82	0.7	0.75	930
o	0.88	0.8	0.84	985
P	0.95	0.85	0.9	925
Q	0.68	0.8	0.73	930
R	0.58	0.83	0.68	920
S	0.98	0.76	0.86	970
T	0.88	0.84	0.86	945
U	0.97	0.57	0.72	930
V	0.49	0.99	0.66	920
W	1	0.64	0.78	930
X	0.81	0.83	0.82	825
Y	0.25	0.27	0.26	1000
Z	1	0.74	0.85	890
accuracy			0.76	24594
macro avg	0.8	0.77	0.77	24594
weighted avg	0.8	0.76	0.77	24594

Figure 7.16: Model 4 report on 30% Training data(Dataset 1) & Unseen data (Dataset 2)

The results presented in Figure 7.16 of the report provide strong evidence that the inclusion of a more varied training dataset has significantly improved the feature learning capabilities of the model. This is reflected in the model's training accuracy of 89%, which is an improvement over the previous models. Moreover, the model was able to correctly identify 77% of the classes in the unseen testing dataset, indicating that the model has achieved better generalization. The increase in the number of layers from the previous models to seven has allowed for more complex feature extraction, which is evident in the improved performance of the model. Additionally, the model's ability to learn more complex features has been facilitated by the increase in the number of learning parameters, which has enabled the model to better capture the nuances of the data. Overall, these results indicate that the use of a more varied training dataset, combined with an increased number of layers and learning parameters, has significantly enhanced the model's performance.

7.6 Summarized test result of Models 1-4

The figure 7.17 bar plot summarizes the test results of the four models on both the 30% training split and the unseen data. The figure highlights that modifying the preprocessing steps can enhance the quality of learnable features, while tuning the model parameters and adding variation to the training dataset can improve the model's learning performance.

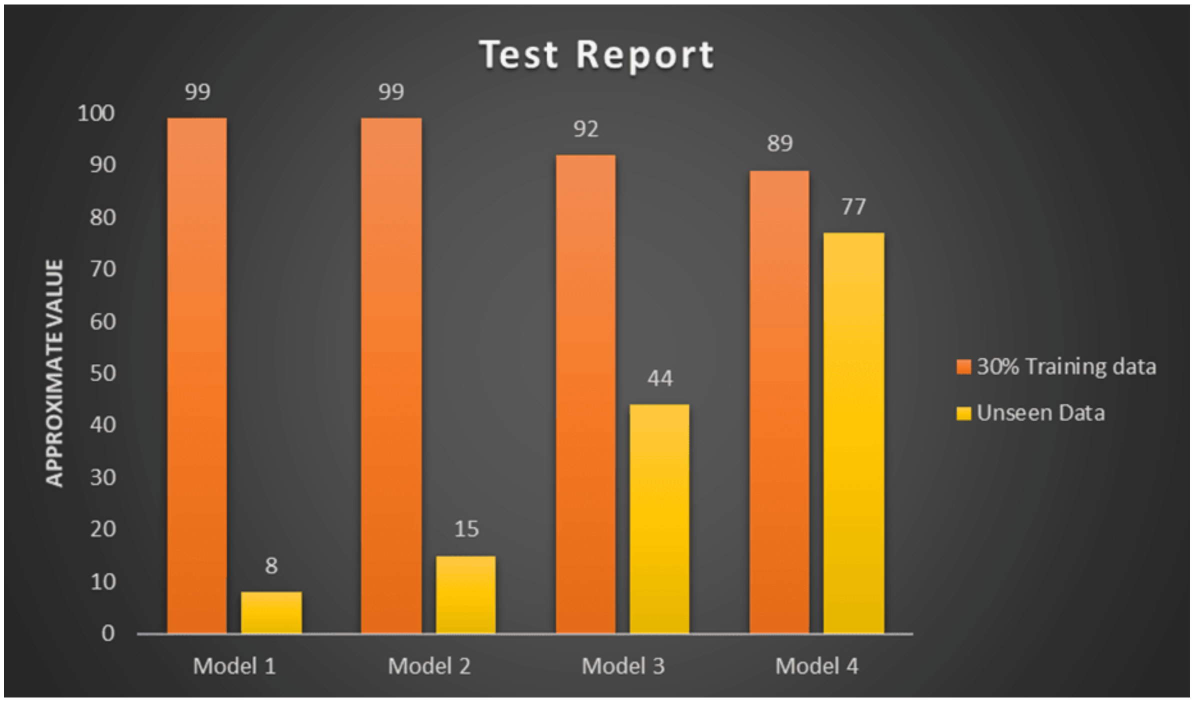


Figure 7.17: Test report summary

However, none of the four models are found to be effective or satisfactory in predicting the class of signs in live video frames, due to several reasons. The models are trained on static image datasets, while video frames are a continuous stream of image frames, leading to poor performance when applied to live video data. In some cases, the preprocessing approaches used are unable to remove the noise created by different lighting conditions and backgrounds, which negatively impacts the model's accuracy. Furthermore, it is crucial to note that the dataset used to train the model may not always reflect the real-world environment.

7.7 Model 5

Preprocessing:

- Mediapipe hand detection
- Drawing finger joints
- Gaussian Blur
- Gray scaling
- Edge detection

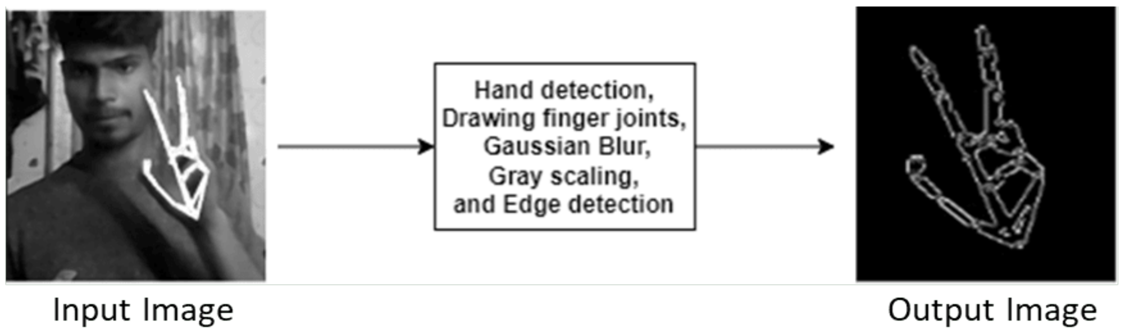

Input Image
Hand detection, Drawing finger joints, Gaussian Blur, Gray scaling, and Edge detection
Output Image

Figure 7.18: Transformed image

The Model 1 - 4 employed a CNN model architecture and a limited range of training data, which led to several limitations. The models' preprocessing techniques are inadequate in removing noise in varying environmental conditions, resulting in inconsistent performance. To overcome these limitations, we adopted Google's Mediapipe hand solution API. This API detects hand regions and overlays the extracted finger key points on a 3-channel black image, effectively reducing background noise. This allowed the model to accurately identify hand positions in any given frame and extract hand regions where the hand signs were performed shown in figure 7.18. This approach significantly improved the model's performance by eliminating the dependence on the CNN model architecture and enabling the extraction of complex hand features, leading to more accurate and reliable results.

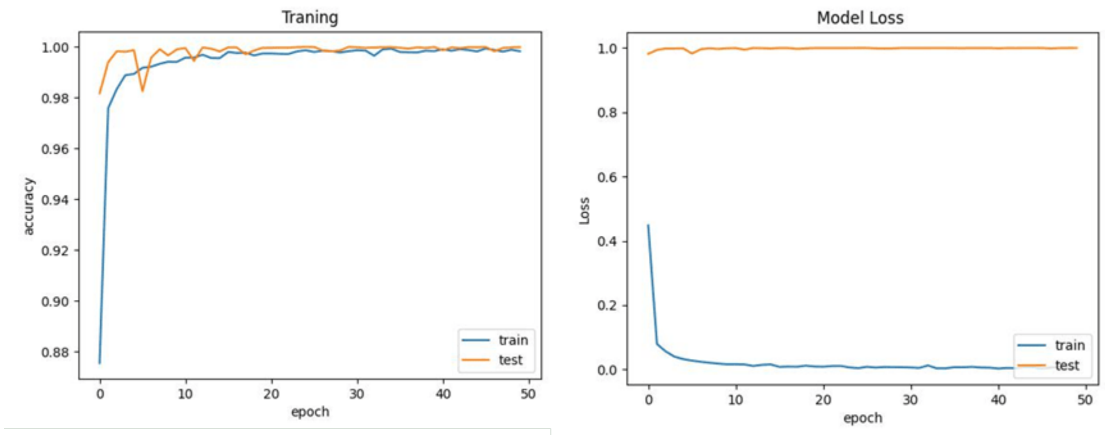
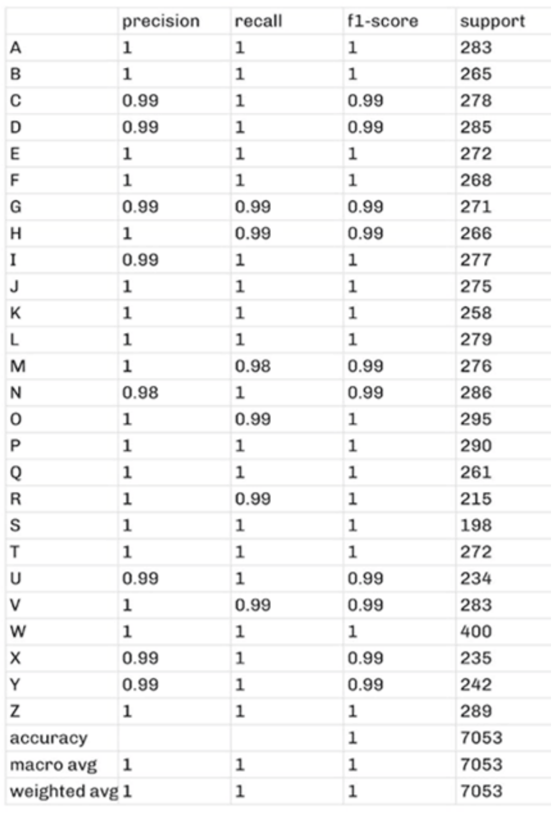


Figure 7.19: Model 5 training results



	precision	recall	f1-score	support
A	1	1	1	283
B	1	1	1	265
C	0.99	1	0.99	278
D	0.99	1	0.99	285
E	1	1	1	272
F	1	1	1	268
G	0.99	0.99	0.99	271
H	1	0.99	0.99	266
I	0.99	1	1	277
J	1	1	1	275
K	1	1	1	258
L	1	1	1	279
M	1	0.98	0.99	276
N	0.98	1	0.99	286
O	1	0.99	1	295
P	1	1	1	290
Q	1	1	1	261
R	1	0.99	1	215
S	1	1	1	198
T	1	1	1	272
U	0.99	1	0.99	234
V	1	0.99	0.99	283
W	1	1	1	400
X	0.99	1	0.99	235
Y	0.99	1	0.99	242
Z	1	1	1	289
accuracy			1	7053
macro avg	1	1	1	7053
weighted avg	1	1	1	7053

Figure 7.20: Model 5 report on unseen set

The results in figure 7.19 & figure 7.20 illustrates that the model achieved an accuracy of 99.50% on both the training and validation datasets.

This indicates that the model was able to generalize well and accurately classify the input data. Due to the significant variation in the prediction of CNN Model 5, when applied directly to real-time video frames, it was deemed necessary to adopt a transfer learning technique to integrate the model into our application. We employed K-Nearest Neighbor (KNN) classification with three neighbors to achieve this goal. The KNN model was trained on the output of Model 5, and the performance of both CNN-Model 5 and KNN was assessed using 30% of unseen data from dataset 3, which served as a separate testing set. By utilizing this approach, we are able to achieve 99.99 % accuracy shown in figure 7.21. A more stable and accurate classification of class labels in real-time video frames.

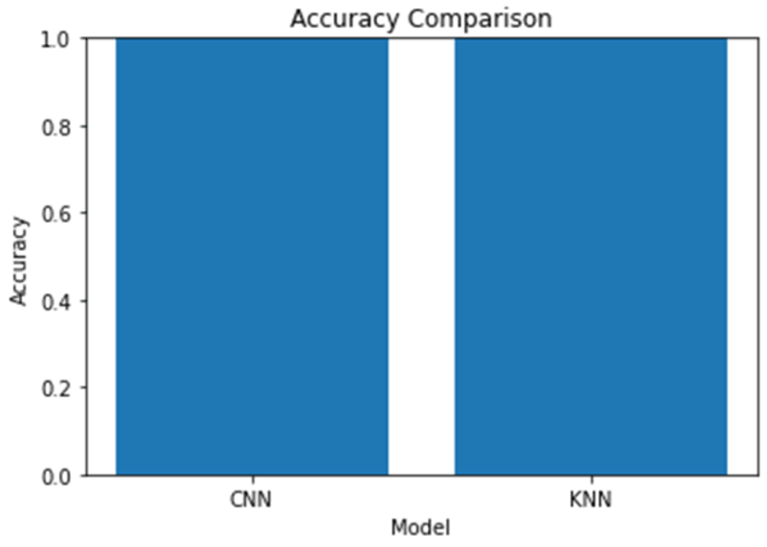


Figure 7.21: CNN and KNN (Model 5) result comparison

7.8 Text to Sign Language Animation

The process of presenting animations based on text input comprises several steps. Firstly, the input text undergoes a preprocessing stage using the Natural Language Toolkit (nltk), which filters out words based on the animation signs that are available on the server. The filtered text signs animations are then played on the application's frontend illustrated in figure 7.21.

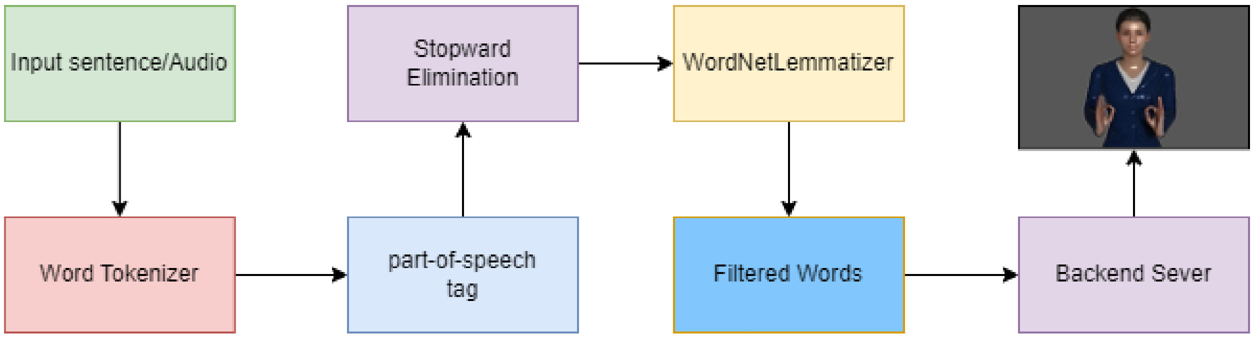


Figure 7.22: Text to ISL Animation

To create animations for Indian Sign Language hand motions, the team utilized Blender, a 3D modeling software. Mesh characters were designed and rigged with bones to enable movement. Finally, a rig control was added to the character, allowing for precise control over its movements as shown in figure 7.23.

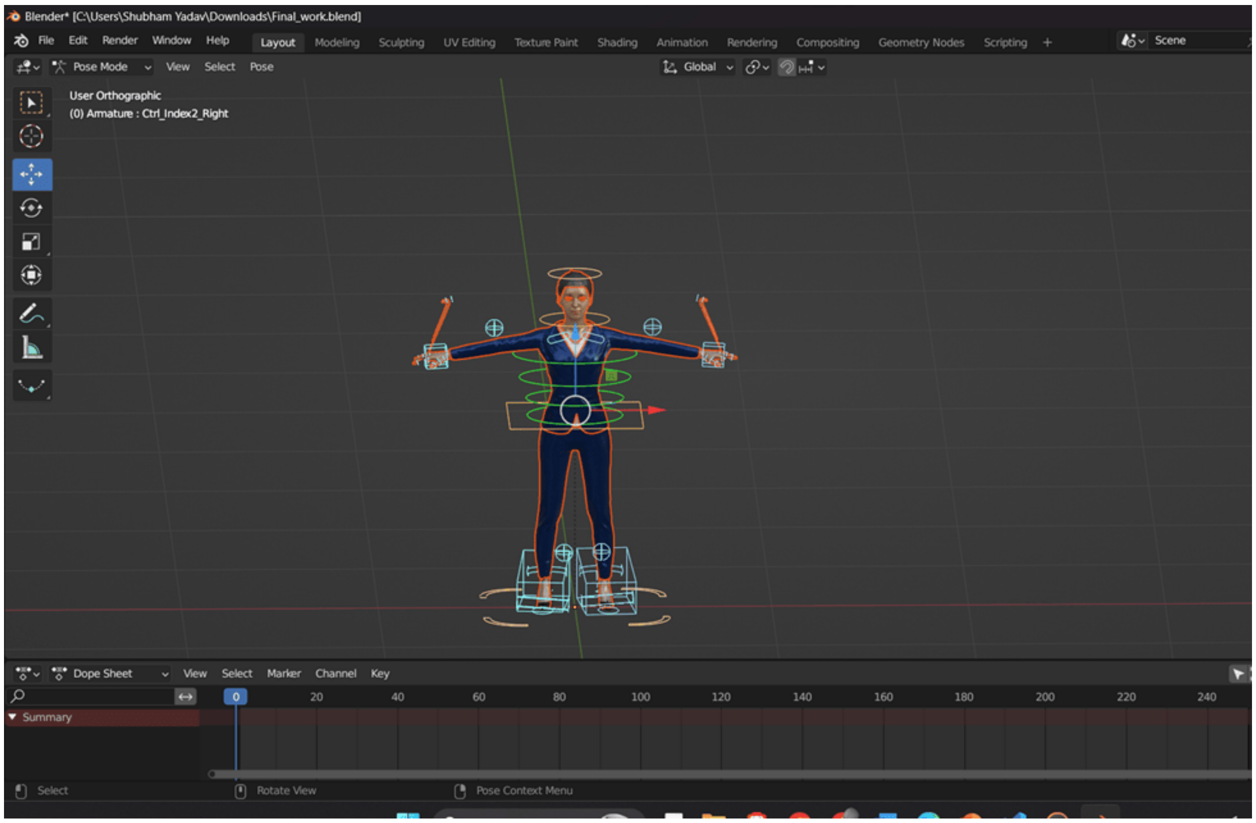
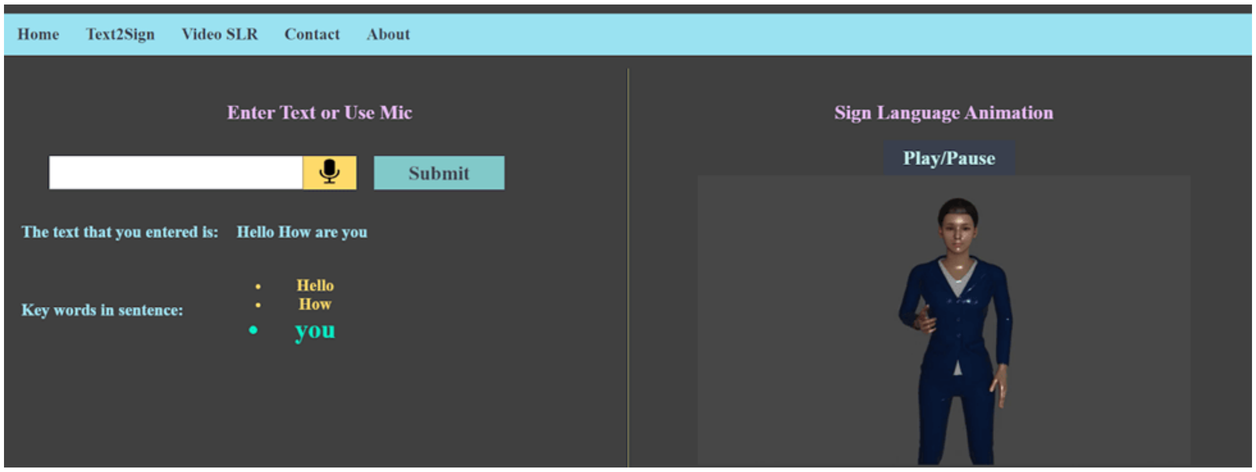

Blender* [C:\Users\Shubham Yadav\Downloads\Final_work.blend]
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Scene
Pose Mode View Select Pose
Global
User Orthographic
(0) Armature : Ctrl_Index2_Right
Dope Sheet View Select Marker Channel Key
Summary
Select
Rotate View
Pose Context Menu

Figure 7.23: Blender 3D character used for animation

Blender is chosen for its advanced capabilities in 3D modeling and animation, including the ability to create detailed mesh structures and rigging systems. The software is widely used in the industry and has an active community of developers and users, making it a reliable choice for complex animation projects.



The text that you entered is: Hello How are you

Key words in sentence:
- Hello
- How
- you

Figure 7.24: Animation playing on frontend

The use of Blender and rigging systems allowed for the creation of realistic and accurate animations of Indian Sign Language hand motions, which are integrated into the application's user interface. This feature enables users to communicate with Sign Language in a more natural and intuitive way, without requiring prior knowledge of the language shown in figure 7.24.

Table 7.1: Model approch and result summary of vision based recognition

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Pre-processing Approach	Gaussian Blur and Gray Scaling	Gaussian Blur, Gray scaling, Thresholding and Edge detection	Gaussian Blur, Gray scaling, Edge detection, Varying Augmentation parameters and 30% Dataset 2 mixture.	Gaussian Blur, Gray scaling, Edge detection and 70% of Dataset 1 + 70% of Dataset 2	Mediapipe hand detection, Drawing finger joints, Gaussian Blur, Gray scaling and Edge detection
CNN Layers	6	6	6	6	6
Activation, Optimizer	ReLU, Adam	ReLU, Adam	ReLU, Adam	ReLU, Adam	ReLU, Adam
Result	Accurate classification of test classes Low recognition rate of 0.8% on unseen data Inability to recognize in real-time video frames Model needs improvement for practical use	Accurate classification of all test set classes Improved recognition rate from 0.8% to 15% on unseen data Indicates enhanced generalization capabilities Still struggles to recognize hand signs in real-time video frames.	30% of Dataset 2 included in training set Recognition rate improved on unseen set Improvement observed Dataset 2 incorporation helped	Model achieved 77% accuracy on unseen test dataset Poor performance in real-time classification Unable to identify classes in real-time Further improvements required for real-time usage.	Model achieved 99.50% accuracy. Real-time sign language recognition. K-Nearest Neighbor classifier used. Effective communicati-on for deaf individuals.

Chapter 8

RESULTS

8.1 Result

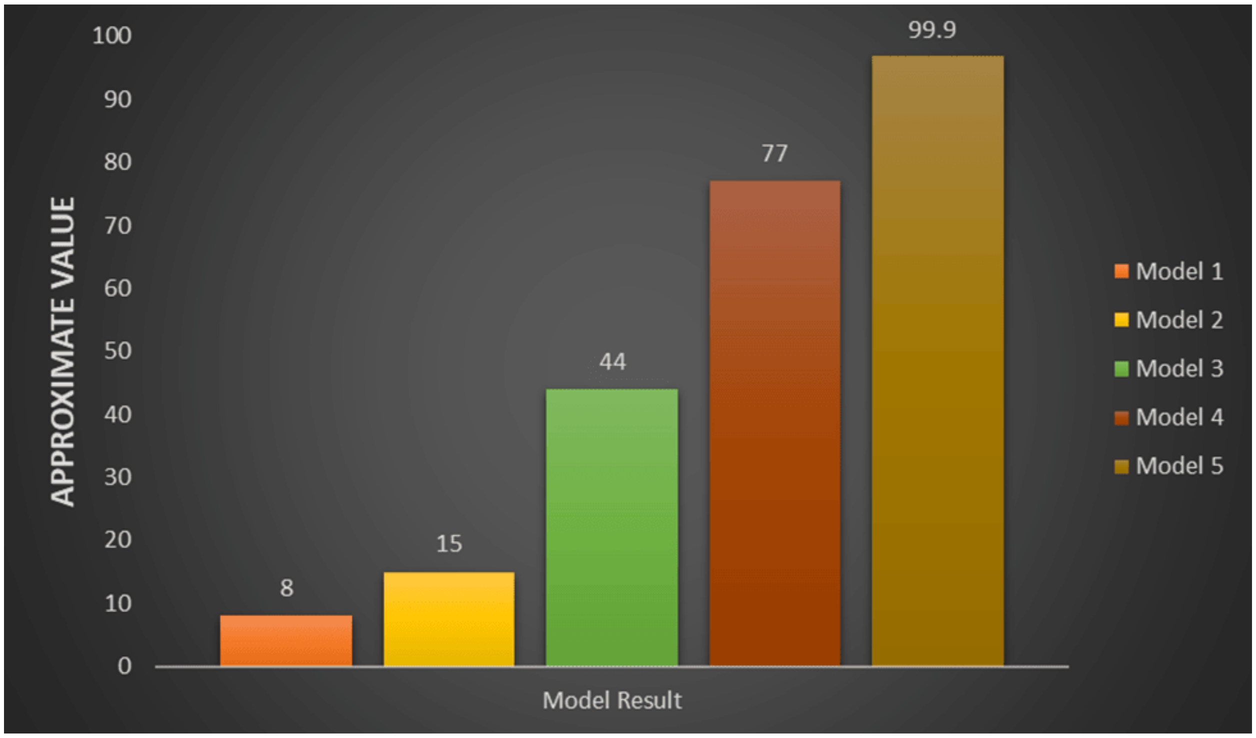


Figure 8.1: Model 1-5 results

The hand sign recognition models underwent evaluation on 30% of unseen data that is separated from the training set. The evaluation revealed that adjusting the preprocessing steps, enhanced performance on static data, although the model's performance degraded significantly when tested on real-time video frames. This could be attributed to the fact that preprocessing steps used for static images may not be suitable for real-time videos. To overcome this issue, a different approach is implemented, which involves creating a new dataset by capturing frames of hand signs. This approach resulted in a significant improvement in the model's performance result included in figure 8.1, enabling better identification of hand signs in real-time video frames. This emphasizes the importance of developing models that are relevant to the target application. In this case, the requirement to identify hand signs in real-time video frames is recognized, and a new dataset and corresponding approach are developed to meet this need.

Chapter 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

To build a model for real-time applications, careful attention should be given to preprocessing and feature extraction. Our experiments demonstrated that Model 5 outperformed previous models in real-time applications. It is observed that modifying the preprocessing steps improved the results on static data but resulted in poor performance in real-time applications. Thus, we recommend extracting features from video frames using model 5 approach or other techniques like optical flow and training the model using more complex neural network structures such as CNN-RNN to obtain better results.

9.2 Future Scope

The future scope involves leveraging advanced techniques such as CNN-RNN and temporal feature learning to identify signs from a sequence of frames. Further development of sign language recognition systems may involve presenting sign language animations from text input, which requires additional text processing and the collection of Indian sign language animations to ensure a smooth conversation flow. This interface can be integrated into various devices such as smartphones or tablets, which would make it easily accessible to the deaf and hard of hearing community.

Chapter 10

REFERENCE

Reference

[1] K.Shenoy,Tejas Dastane,Varun Rao,Devendra Vyavaharkar,“Real-timeIndianSignLanguage(ISL)Recognition” ,1 July 2021.

[2] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, Benjamin Schrauwen. ”Sign Language Recognition using Convolutional Neural Networks”, 2015

[3] G.Anantha Rao, K.Syamala, P.V.V.Kishore, A.S.C.S.Sastry, “Deep Convolutional Neural Networks for Sign Language Recognition.” , 2018.

[4] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, Weiping Li, “Video-Based Sign Language Recognition without Temporal Segmentation”, AAAI-18, April 2018

[5] M.A Hossen, Arun Govindaiah, Sadia Sultana, Alauddin Bhuiyan “Bengali Sign Language Recognition UsingDeep Convolutional Neural Network” June 2018

[6] Kshitij Bantupalli, Ying Xie, “American Sign language identification using Deep Learning and Computer Vision.”, IEEE,2018.

[7] Sruthi C. J and Lijiya A. Member, “Signet: A Deep Learning based Indian Sign Language Recognition System.”,IEEE, April 2019.

[8] Lean Karlo S. Tolentino, Ronnie O. Serfa Juan, August C. Thio-ac, Maria Abigail B. Pamahoy, Joni Rose R. Fortezaz and Xavier Jet O. Garcia. “Sign language identification using Deep Learning.”, IJMLC, December 2019.

[9] Ankita Wadhawan, Parteek Kumar, “Deep learning-based sign language recognition system for static signs” , Jan 2021

[10] Prof. Radha S. Shirbhate, Mr. Vedant D. Shinde, Ms. Sanam A. Metkari, Ms. Pooja U. Borkar, Ms. Mayuri A. Khandge. ”Sign language Recognition Using Machine Learning Algorithm” March 2020

[11] Dongxu Li , Chenchen Xu, Xin Yu, Kaihao Zhang, Ben Swift , Hanna Suominen, Hongdong Li, ”TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation” 2020

[12] Necati Cihan Camg, Oscar Koller, Simon Hadfifield and Richard Bowden, “Sign Language Transformers: JointEnd-to-end Sign Language Recognition and Translation”, 2020.

[13] Rachana Patil, Vivek Patil, Abhishek Bahuguna, and Mr. Gaurav Datkhile. ”Indian Sign Language Recognition using Convolutional Neural Network” 2021

[14] Razieh Rastgoo, Kourosh Kiani, Sergio Escalera, “Word separation in continuous sign language using isolated signs and post-processing”, 13 Apr 2022.

[15] Abdul Mannan , Ahmed Abbasi , Abdul Rehman Javed , Anam Ahsan ,Thippa Reddy Gadekallu , and Qin Xin “Hypertuned Deep Convolutional Neural Network for Sign Language Recognition”,30 April 2022.

[16] Suyog Bhamare, Abhishek Shinde, Rahul Palve, A. N. Gharu. ” Sign Language Detection with CNN”,May 2022

[17] Deep Kothadiya 1, Chintan Bhatt 1, Krenil Sapariya , Kevin Patel, Ana-Belén Gil-González and Juan M. Corchado ”Deepsign: Sign Language Detection and Recognition Using Deep Learning”,June 2022

[18] Md. Nafis Saiful, Abdulla Al Isam, Hamim Ahmed Moon, Rifa Tammana Jaman, Mitul Das, Md. Raisul Alam, Ashifur Rahman. ”Real-Time Sign Language Detection Using CNN”, Oct 2022

[19] Ahmed Sultan, Walied Makram, Mohammed Kayed, Abdelmaged Amin Ali. ”Sign language identification and recognition: A comparative study”, 2022

[20] Sharvani Srivastava, Amisha Gangwar, Richa Mishra, Sudhakar Singh. ”Sign Language Recognition System using TensorFlow Object Detection API”,2022

[21] Prem Selvaraj, Gokul NC, Pratyush Kumar, Mitesh Khapra. ”OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages”, 2022

[22] Vaishnavi Sonawane,”Indian Sign Language Dataset” https://www.kaggle.com/datasets/vaishnaviasonawane/indian-sign-language-dataset October 2020.

[23] ATHARVA DUMBRE,”Indian Sign Language (ISLRTC referred)” https://www.kaggle.com/datasets/atharvadumbre/indian-sign-language-islrtc-referred May 2022.

[24] https://www.blender.org/.

[25] Martín A., Paul B., Jianmin C., Zhifeng C., Andy D., Jeffrey D., Matthieu D., Sanjay G., Geoffrey I., Michael I., Manjunath K., Josh L., Rajat M., Sherry M., Derek G. M., Benoit S., Paul T., Vijay V., Pete W., Martin W., Yuan Yu, and Xiaoqiang Z, Google Brain, ”TensorFlow: A System for Large-Scale Machine Learning” https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi November 2016.

[26] Kukil,”Introduction to MediaPipe” https://learnopencv.com/introduction-to-mediapipe/ January 2023.

[27] Bird, Steven, Edward Loper and Ewan Klein,"Natural Language Processing with Python" O'Reilly Media Inc. https://www.nltk.org/ 2009.

[28] Meena Ugale, Rohit Nalawde, Apoorv Nagap, Lakhan Jindam "Agriculture Field Monitoring and Plant Leaf Disease Detection", 3rd International Conference-IEEE, CSCITA-2020, doi:10.1109/CSCITA47329.2020. 9137805 April 2020.

[29] Gonsalves, T.,Upadhyay, J., "Integrated deep learning for self-driving robotic cars",Artificial Intelligence for Future Generation Robotics,2021, pp. 93–118.

[30] https://developersbreach.com/convolution-neural-network-deep-learning/

[31] https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Chapter 11

APPENDIX

Gmail **Odrin Rodrigues <odrinrodrigues@gmail.com>**

ICAMIDA 2022 notification for paper 2454

ICAMIDA 2022 <icamida2022@easychair.org> 7 November 2022 at 17:09
To: Odrin Rodrigues <odrinrodrigues@gmail.com>

Dear Author(s) Odrin Rodrigues, Shivam Yadav, Anushka Shinde, Kaustubh Desle, Meena Ugale

We are happy to mention that your paper, as per the following details:

Paper ID: 2454

Title: A Review on Sign Language Recognition using CNN

has been accepted for the conference.

Please note the following regarding your paper:

You will have to format the paper as per the Springer Conference Format, either in Word or LaTeX format (see https://www.springer.com/gp/authors-editors/conference-proceedings/conference-proceedings-guidelines). For LaTeX related files, you will need to send us all the relevant files along with the compiled version.

The report of the reviewers is given here. As per the report, you will have to modify the paper with minor / major revisions.

You are requested to register for the conference. The link for payment and registration is given here (see https://erp.mgmu.ac.in/asd_EventPublicUserMaster.htm?eventID=42).

You would need to email us your ID proof, at the email id icamida@mgmu.ac.in, with the subject line - "[ICAMIDA – ID – Proof] Paper ID no ____"

Please complete the registration process, submission of the paper in the correct format, and sending of the Proof – ID within the next 4 to 5 days.

N.B. Please note that although your paper has been accepted for the conference, it would undergo thorough quality checks with Springer Nature (our publishers for the proceedings). Please make sure that you have obtained permissions for all the third-party material used in your papers, and you have the relevant documentation, regarding the same. We would approach you with another form regarding this.

NOTE: The revised paper should have less than 15 % similarity index on the Springer Nature's tool.

Kindly share your contact number with the organizers for further communication.

ICAMIDA-2022

Organizing committee

SUBMISSION: 2454
TITLE: A Review on Sign Language Recognition using CNN

----------------------- REVIEW 1 ---------------------
SUBMISSION: 2454
TITLE: A Review on Sign Language Recognition using CNN
AUTHORS: Odrin Rodrigues, Shivam Yadav, Anushka Shinde, Kaustubh Desle and Meena Ugale

----------- Overall evaluation -----------
SCORE: 2 (accept)
----- TEXT:
Hand gesture based devices are already in proto level. Please mention advantages.

A Review on Sign Language Recognition Using CNN

Meena Ugale( ), Odrin Rodrigues Anushka Shinde, Kaustubh Desle, and Shivam Yadav

Department of Information Technology, Xavier Institute of Engineering, University of Mumbai, Mumbai, Maharashtra, India
meena.u@xavier.ac.in

Abstract. In sign language, hand gestures are used as one type of non-verbal communication. Individuals with hearing or speech problems typically use it to communicate with others or among themselves. Many makers around the world have created numerous sign language systems. The software that shows a system prototype capable of automatically recognizing sign language to assist deaf and dumb individuals in communicating with each other or regular people more successfully. The study demonstrates that there is ongoing research in the field of vision-based hand gesture recognition, with various studies being undertaken and a large number of publications appearing every year in journals and conference proceedings. Data acquisition, data environment, and hand gesture representation are the three main areas of concentration in publications on the hand gesture recognition system. In terms of recognition precision, we have also analyzed how well the recognition system performs. The recognition accuracy for the signer dependent spans from 69% to 98%, with an average of 88.8% among the chosen experiments.

Keywords: Sign Language Recognition · Communication · Data Acquisition · Convolution Neural Network · Deep Learning

1 Introduction

The lack of communication between people with hearing or speaking impairments and the rest of the world. This can be frustrating and isolating for people with hearing or speaking impairments. When spoken communication is impossible or undesirable, sign language is used to communicate through bodily movements, particularly those of the hands and arms as shown in Fig. 1. Though sign languages are very effective in communication, they are not commonly used or known. This creates a communication barrier.

The American Sign Language (ASL), British Sign Language (BSL), Indian Sign Language (ISL), and others are all different sign languages. Similar to how spoken languages have a vocabulary of words, sign languages too have a vocabulary of signs. The grammar of sign languages varies from country to country

S. Tamane et al. (Eds.): ICAMIDA 2022, ACSR 105, pp. 251–259, 2023.
https://doi.org/10.2991/978-94-6463-136-4_23

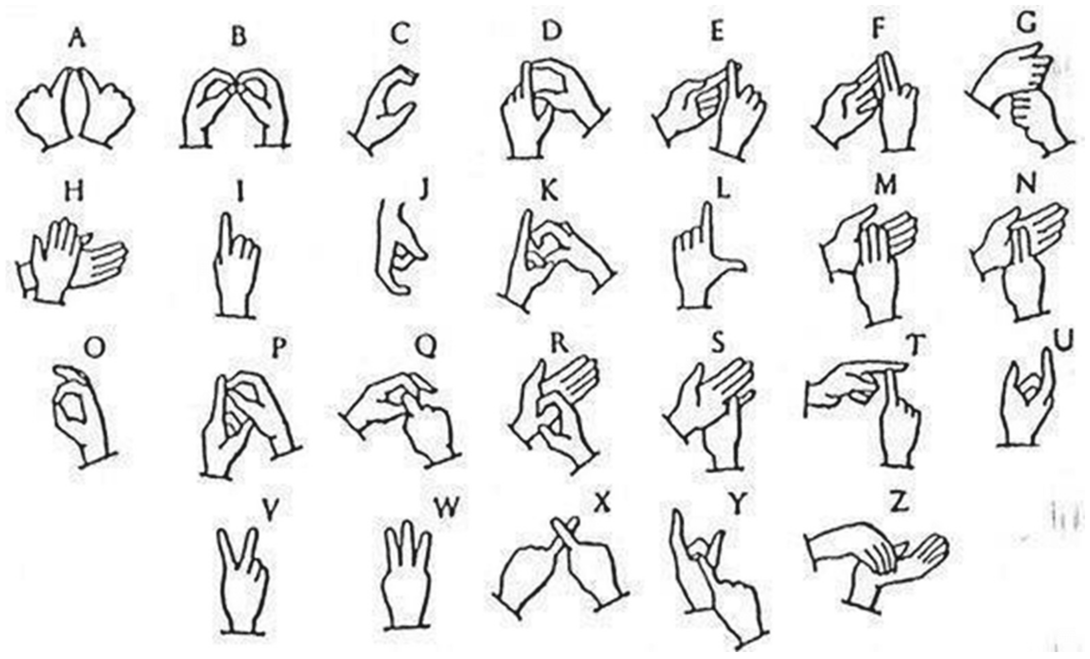


Fig. 1. Alphabetic Hand Sign [11].

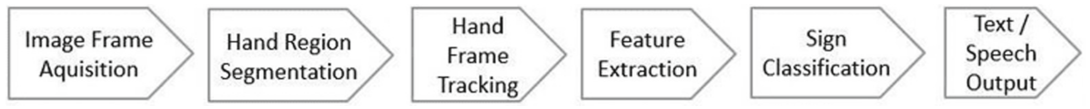


Fig. 2. Block Diagram of Sign Language Recognition.

and is not standardized or universal. A manual sign language interpreter is not always a good idea and frequently intrudes on the subject's right to privacy. This problem can be resolved by using an automated sign language translator that can translate sign language into spoken or written language as shown in Fig. 2. The hearing- and vocally- impaired people will benefit from an accurate automatic sign language translator since it will allow them to live independently and in close contact with others for the rest of their lives.

2 Convolution Neural Network Concept

Machine learning includes convolutional neural networks (CNNs). It is a subset of the several artificial neural network models that are employed for diverse purposes and data sets. A CNN is a particular type of network design for deep learning algorithms that is utilized for tasks like image recognition and pixel data processing [13,14]. The structure of a CNN is comparable to the connection structure of the human brain. Similar to how the brain has billions of neurons, CNNs also have neurons, but they are structured differently. A CNN performs better with image inputs and voice or audio signal inputs compared to the earlier networks.

2.1 CNN Layers

A deep learning CNN is composed of three layers: a convolutional layer, a pooling layer, and a fully connected (FC) layer. The first layer is the convolutional layer, while the final layer is the FC layer. The complexity of the CNN grows from the convolutional layer to the FC layer. The CNN is able to identify increasingly larger and more intricate aspects of an image until it successfully recognizes the complete thing as a result of the rising complexity.

Convolution Layer: The convolutional layer, the central component of a CNN, is where most computations take place. The first convolutional layer may be followed by a subsequent convolutional layer. A kernel or filter inside this layer moves over the image's receptive fields during the convolution process to determine whether a feature is present. The kernel traverses the entire image over a number of iterations. A dot product between the input pixels and the filter is calculated at the end of each iteration. A feature map or convolved feature is the result of the dots being connected in a certain pattern. In this layer, the image is ultimately transformed into numerical values that the CNN can understand and extract pertinent patterns from.

Pooling Layer: The pooling layer similarly to the convolutional layer sweeps a kernel or filter across the input image. Contrary to the convolutional layer, the pooling layer has fewer input parameters but also causes some information to be lost. Positively, this layer simplifies the CNN and increases its effectiveness.

Fully Connected Layer: Based on the features extracted in the preceding layers, picture categorization in the CNN takes place in the FC layer. Fully connected in this context means that every activation unit or node of the subsequent layer is connected to every input or node from the preceding layer. The CNN does not have all of its layers fully connected because that would create an excessively dense network. It would cost a lot to compute, increase losses, and have an impact on output quality.

2.2 Working

As shown in Fig. 3 each layer trains the CNN to recognize the many aspects of an input image. Each image is given a filter or kernel to create an output that gets better and more detailed with each layer. The filters may begin as basic characteristics in the lower layers. In order to check and identify features that specifically reflect the input item, the complexity of the filters increases with each additional layer. As a result, the partially recognized image from each layer's output, or convolved image, serves as the input for the subsequent layer. The CNN recognizes the image or object it represents in the final layer, which is an FC layer. The input image is processed through a number of different filters during convolution. Each filter performs its function by turning on specific aspects of

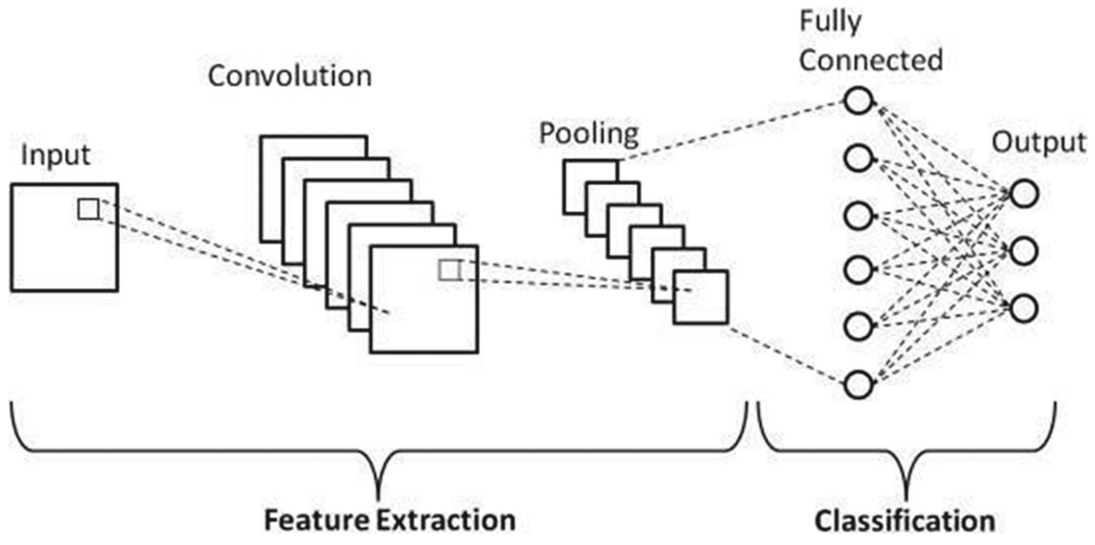


Fig. 3. Convolution Neural Network Architecture [12].

the image, after which it sends its output to the filter in the subsequent layer. The operations are repeated for dozens, hundreds, or even thousands of layers as each layer learns to recognize various features. Finally, the CNN is able to recognize the full object after processing all the picture data through its many layers.

3 CNN-Based Approach for Sign Language Recognition

According to paper [1], the model is constructed with input layer, four convolutional layers, five rectified linear units (ReLu), two stochastic pooling layers, one dense and one SoftMax output layer. The CNN design employs convolutional layers with various pooling sizes, an activation function and a rectified linear unit handeling non-linearities. Through supervised learning, the network is trained to learn the characteristics of each symbol. By switching from ANN to deep ANN, the identification accuracy is further enhanced, with a claimed 5% improvement in recognition rate. Hence, CNN's are a suitable tool for simulating sign language recognition on mobile platforms [1].

In proposed paper [2], Hierarchical Attention Network with Latent Space (LS-HAN) is used to translate signing videos sentence-by-sentence. By using a sliding window method, each video is split into frame segments. Upon encoding, the Hierarchical Attention Network (HAN) is given the start symbol "#Start," which marks the start of sentence prediction. For each decoding timestamp, the word with the highest probability after the soft max is selected as the predicted word, and its representation is sent to (HAN) for each succeeding timestamp until end flag "#End" is raised [2].

The author [3] approach was designed to function with one-handed gestures. Convolution, Max-Pooling, ReLU, Dropout, Fully Connected, and SoftMax layers form up the deep learning network. The neural network employs a stack of layers to perform classification using the features that are collected from the convolution layers. Using the pre-trained network, the features required for classification are extracted once from the dataset. The validation accuracy of the authors' proposed model was 84.68%, with a validation loss of 0.3523 [3].

Kshitij Bantupalli, Ying Xie [4] mentioned in the paper about CNN model extracted temporal features from the frames which was used further to predict gestures based on sequence of frames. Two methods were employed to classify the signs: a. using outputs from the Softmax layer, and b. derived the results from the Global Pool layer. The global pool results in a 2048-sized vector, which allowed features to be analyzed by the RNN. These characteristics are transferred to Long-Short-Term Memory (LSTM), allowing for larger time dependencies. The vanishing/exploding gradient problem is handled with LSTMs, enabling improved accuracy on larger data sets. On the training set, the model scored high accuracy of 99%. The LSTM utilized sequence data to classify the gesture segments that CNN recognised and processed into one of the gesture classes. The system was able to achieve 93% accuracy with Softmax Layer rather than Pool Layer [4].

The paper [5] presented a vision based deep learning architecture for signer independent Indian sign language recognition system. Total 24 ISL static alphabets were trained using CNN, with training accuracy of 99.93% and testing and validation accuracy of 98.64%. The acquired recognition accuracy exceeds the majority of the techniques [5].

The author of paper [6] proposed a model where the network uses a Stochastic gradient descent optimizer as its optimizer to train the network having a learning rate of 1×10^{-2} . With a batch size of 500, the network was trained over the period of 50 epochs. The image size for training and evaluation was (50, 50, 1). To improve outputs, the Keras and CNN architecture is used, which comprises a number of layers for data processing and training. The CNN layers included more 64 filters. The fully connected layer is being specified by the dense layer along with rectified linear activation [6].

Ankita Wadhawan, Parteek Kumar [7] proposed sign language recognition system includes four major phases that are data acquisition, image preprocessing, training and testing of the CNN classifier. The model training is based upon convolutional neural networks. Preprocessed sign pictures were fed into the classifier, which assigns them to the appropriate category. The dataset of several ISL gestures is used to train the classifier. The system achieved training and validation accuracy of 99.76% and 98.35%, respectively, using RMSProp and it has been found that the SGDoptimizer outperformed Adam, RMSProp and other optimizers with training and validation accuracy of 99.90% and 98.70%, respectively, on gray scale image dataset [7].

The paper [8] used a modified version of JoeyNMT to implement the Sign Language Transformers. All the components of network were built using the PyTorch. 8 heads in each layer and 512 hidden units were used to construct the framework. Used a batch size of 32 to train the networks using the Adam optimizer. Network is evaluated at every 100 iterations [8].

Boundary identification of the sign presented the biggest hurdle in CSLR. The paper indicates Use of transfer learning to solve this issue. The authors developed a two-step solution model and a post- processing methodology. The hand-crafted SVD utilized to feed features to LSTM Network extracted from the features using the prediction model. Both the SVD feature extractor and the hand pose estimator receives a window size of 50 frames. Then, the many-to-one LSTM Network maps the matching SVD feature sequence. Into a single vector up to 50 frames. After that, a Fully Connected (FC) layer receives this vector. Finally, the FC outputs are covered with a Softmax layer. The suggested methodology employed a specified threshold, 0.51, to approve or reject a recognized class for the separation of the isolated signs in a continuous sign video. There are various difficulties with the comparable signs. In the sliding window that is open. There were some difficulties in the placards that said "Congratulation", "Excuse", "Upset", "Blame", "Fight" and "Competition". In order to learn more powerful features to better describe sign categories and decrease miss-classifications as a result, adding more samples could to title t inter-class variation [9].

Paper suggested an architecture based on CNN, containing dense, max-pooling, dropout, and many convolutional layers (fully-connected). Which performs convolution on input with various filter and kernel sizes to map feature. Model correctly learns features from three main blocks, each of which has a different parameter configuration and uses ReLU as an activation function. The flattening layer transforms input into a vector before connecting to a group of the fully connected layer. Authors evaluated the model on new data in a later stage and reported accuracy of 99.67% [10] (Table 1).

4 Discussion and Conclusion

Numerous authors with expertise in the deep learning field offered novel approaches to the Sign Language Recognition challenge. The dataset is the most crucial prerequisite for a sign language recognition system. On the internet, there exist numerous datasets for various sign languages, including ASL, American Sign Language, CSL, etc. For training and testing, authors [3] [4] [5] [6] [7] manually created their own dataset on the system. When implementing a model with a pooling layer, authors in [4] experienced poor accuracy (58%), authors in [9] encountered conflicts with other signs having the same hand movements, resulting in the wrong classification, and authors in [8] used a pre-trained CNN+LSTM+HMM setup followed by RELU function. To achieve good accuracy, the majority of the authors seeded their input layer using extracted features and retrained the model on average over 38–45 iterations. Among the other proposed systems for static signs, the author [7] achieved the best performance accuracy of 99.90%. The Softmax layer [4] technique produced a CSLR accuracy of 93%.

Table 1. Summary of CNN Based Approaches Applied for Sign Language Recognition.

Year	Authors	Title	Architecture	Performance/ Results
2019	G.Anantha Rao, K.Syamala, P.V.V.Kishore, A.S.C.S.Sastry [1].	"Deep Convolutional Neural Networks for Sign Language Recognition"	CNN	Models average recognition rate was 92.88%.
Apr 2018	Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, Weiping Li [2].	"Video-Based Sign Language Recognition without Temporal Segmentation"	Two-stream 3D CNN	Proposed method LS-HAN achieves 82.7 % accuracy which is more than LSTM-E 76.8%.
June 2018	M.A Hossen, Arun Govindaiah, Sadia Sultana, Alauddin Bhuiyan [3].	"Bengali Sign Language Recognition Using Deep Convolutional Neural Network"	Pre-trained VGG16, CNN	Recognition rate - validation loss of 0.3523 and validation of 84.68% achieved.
2018	Kshitij Bantupalli, Ying Xie [4].	"American Sign Language Recognition using Deep Learning and Computer Vision"	CNN, RNN, Machine learning, HMM.	Instead of using Pool Layer, the system was able to attain 93% accuracy with SoftMax Layer.
Apr 2019	Sruthi C. J and Lijiya A [5].	"Signet: A Deep Learning based Indian Sign Language Recognition System"	CNN	Training accuracy = 99.93% and validation accuracy was equal to 98.64%.
Dec 2019	Lean Karlo S. Tolentino, Ronnie O. SerfaJuan, August C.Thio-ac, Maria Abigail B.Pamahoy, Joni Rose R. Fortezaz and Xavier Jet O. Garcia [6].	"Sign language identification using Deep Learning"	CNN	The system's accuracy was on average 93.667%. The testing has a 90.04% letter recognition accuracy, a 93.44% number recognition accuracy, and a 97.52% static word identification accuracy.
Jan 2020	Ankita Wadhawan, Parteek Kumar [7].	"Deep learning- based sign language recognition system for static signs"	CNN	A performance accuracy rate of 99.90% was calculated by the authors.
2020	Necati Cihan Camg,Oscar Koller, SimonHadfifield and Richard Bowden [8].	"Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation"	RNN-based attention architectures, Connectionist Temporal Classification (CTC).	Compared to earlier approaches, the authors Language Transformers outperform both their recognition and their translation effectiveness with a 2% reduction in word error rate.
13 Apr 2022	Razieh Rastgoo, Kourosh Kiani, Sergio Escalera [9].	"Word separation in Continuous sign language using isolated signs and post- processing"	CNN, LSTM	The proposed model obtains an average of recognized Softmax outputs equal to 0.98 and 0.59.
30 Apr 2022	Abdul Mannan, Ahmed Abbasi, Abdul Rehman Javed, Anam Ahsan, Thippa Reddy Gadekallu, Qin Xin [10].	"Hypertuned Deep Convolutional Neural Network for Sign Language Recognition"	CNN	On test data the suggested Deep CNN model has a 99.67% accuracy rate when recognising ASL alphabets.

Each research suggested a virtually identical architecture, including image augmentation, feature extraction, and fully connected layers for the subsequent classifier outcomes. This research examined the challenges, advancements, and probable future directions of the vision-based hand gesture recognition system. It seems that the publications we read emphasized the value of data collection, features, and the training data's context. It was also noted that the majority of databases utilised in hand gesture recognition research were from a constrained context, underlining the need for sign language databases that are less constrained and incorporate data from diverse environments. The conclusion of this study is thus more attention needs to be paid to the uncontrolled environment, setting to build vision-based gesture recognition system for practical use. Because it can provide researchers a chance to enhance the system's capability to recognize hand gestures in any form of environment. Data collection is a core procedure that has been stressed and placed in the spotlight of many vision-based hand gesture recognition studies.

References

1. G. Anantha Rao, K. Syamala, P.V.V. Kishore, A.S.C.S. Sastry, "Deep Convolutional Neural Networks for Sign Language Recognition." 2018.
2. Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, Weiping Li, "Video-Based Sign Language Recognition without Temporal Segmentation", AAAI-18, April 2018.
3. M.A Hossen, Arun Govindaiah, Sadia Sultana, Alauddin Bhuiyan "Bengali Sign Language Recognition Using Deep Convolutional Neural Network" June 2018.
4. Kshitij Bantupalli, Ying Xie, "American Sign language identification using Deep Learning and Computer Vision." IEEE 2018.
5. Sruthi C. J and Lijiya A. Member, "Signet: A Deep Learning based Indian Sign Language Recognition System." IEEE, April 2019.
6. Lean Karlo S. Tolentino, Ronnie O. Serfa Juan, August C. Thio-ac, Maria Abigail B. Pamahoy, Joni Rose R. Fortezaz and Xavier Jet O. Garcia. "Sign language identification using Deep Learning." IJMLC, December 2019.
7. Ankita Wadhawan, Parteek Kumar, "Deep learning-based sign language recognition system for static signs", Jan 2021.
8. Necati Cihan Camg, Oscar Koller, Simon Hadfifield and Richard Bowden, "Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation", 2020.
9. Razieh Rastgoo, Kourosh Kiani, Sergio Escalera, "Word separation in continuous sign language using isolated signs and post-processing", 13 Apr 2022.
10. Abdul Mannan, Ahmed Abbasi, Abdul Rehman Javed, Anam Ahsan, Thippa Reddy Gadekallu, and Qin Xin "Hypertuned Deep Convolutional Neural Network for Sign Language Recognition", 30 April 2022.
11. Priyanka Rawat "All about Indian Sign Language", 11 Feb 2021. https://1specialplace.com/2021/02/11/all-about-indian-sign-language/
12. Van Hiep Phung and Eun Joo Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets", 23 October 2019.https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_3368059

13. Meena Ugale, Rohit Nalawde, Apoorv Nagap, Lakhan Jindam "Agriculture Field Monitoring and Plant Leaf Disease Detection", 3rd International Conference-IEEE, CSCITA-2020, April 2020. https://doi.org/10.1109/CSCITA47329.2020.9137805.
14. Gonsalves, T., Upadhyay, J., "Integrated deep learning for self-driving robotic cars", Artificial Intelligence for Future Generation Robotics, 2021, pp. 93–118.