HAND GESTURE RECOGNITION AND TRANSLATION FOR INTERNATIONAL SIGN LANGUAGE COMMUNICATION

A PROJECT REPORT
Submitted By

ABDUL KATHAR.J	**(310120243001)**
AROCKIYADASS.A	**(310120243008)**
NAVEEN RAJ .D	**(310120243026)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ANAND INSTITUTE OF HIGHER TECHNOLOGY

ANNA UNIVERSITY::CHENNAI 600 025

MAY - 2024

ANNA UNIVERSITY::CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"HAND GESTURE RECOGNITION AND TRANSLATION FOR INTERNATIONAL SIGN LANGUAGE COMMUNICATION"** is the bonafide work of **" ABDUL KATHAR. J (310120243001), AROCKIYADASS.A (310120243008), NAVEENRAJ.D (310120243026)"** who carried out the project work under my supervision.

SIGNATURE :	**SIGNATURE :**
Dr.K.Karnavel, M.Tech., Ph.D.,	**Ms.R.SASIREKHA,M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR** **ASSISTANT PROFESSOR**
Department of Artificial intelligence and data science, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603103	Department of Artificial Intelligence and Data science, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603103

Submitted to project and Viva Voce Examination held on ________________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank the almighty for showering his abundant blessings onus to successfully complete the project. Our sincere thanks to, our beloved **"Kalvivallal" Late Thiru T. Kalasalingam, B.Com., Founder** for his keen interest and affection towards us.

Our sincere thanks and gratitude to our **SevaRatna Dr. K.Sridharan,M.Com., MBA., Ph.D., Chairman, Dr. S. Arivazhagi, M.B.B.S., Secretary** for giving us the necessary support during the project work. We convey our thanks to our **Dr.K.Karnavel, M.Tech., Ph.D., Principal** for his support towards the successful completion of this project.

We pay our grateful acknowledgement and sincere thanks to **Professor Dr.K.Karnavel, M.Tech., Ph.D., HOD / AI&DS**, for the good coordinating and for better guidance and constant encouragement in completing this project to make it successful one.

We thank entire **Staff Members** of our department and our friends for helping usby providing valuable suggestions and timely ideas for successful completion of the project.

Last but not the least our **Family Members and Friends** have been a great source of inspiration and strength to us during the course of this project work and our sincere thanks to them.

ABSTRACT

Hand gesture communication serves as a fundamental and instinctive mode of expression, particularly crucial for individuals grappling with hearing impairments. This study presents a groundbreaking real-time method leveraging Convolutional Neural Networks (CNNs) to discern sign language hand gestures, with a specific focus on International Sign Language (ISL). This pioneering approach is geared towards interpreting hand gestures captured through live camera feed, thereby streamlining communication for sign language users. Remarkably, the system not only proficiently identifies ISL gestures but also seamlessly translates them into audible speech, effectively eradicating communication barriers between sign language users and those unfamiliar with the language. This technological leap fosters inclusivity, fostering effortless communication regardless of auditory or verbal capabilities. Rigorous testing under diverse conditions underscores the system's robustness, showcasing an outstanding accuracy rate of 97.85%. This achievement underscores the system's viability even amidst the varying background complexities and lighting nuances, affirming its reliability in real-world applications.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CNN	CONVOLUTIONAL NEURAL NETWORK
RNN	RECURRENT NEURAL NETWORK
ISL	INTERNATIONAL SIGN LANGUAGE

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 SYSTEM OVERVIEW

More than 5% of the world's population is affected by hearing impairment. To overcome the challenges faced by these individuals, various sign languages have been developed as an easy and efficient means of communication. Sign language depends on signs and gestures which give meaning to something during communication/ Researchers are actively investigating methods to develop sign language recognition systems, but they face many challenges during the implementation of such systems which include recognition of hand poses and gestures. Furthermore, some signs have similar appearances which add to the complexity in creating recognition systems. This paper focuses on the sign language alphabet recognition system because the letters are the core of any language. Moreover, the system presented here can be considered as a starting point for developing more complex systems. There are two types of sign language recognition methods namely sensor-based and image-based. The first method is dependent on localized sensors or wearing specific gloves. The main benefit of this method is its ability to provide accurate information about signs or gestures such as the movement, rotation, orientation as well as positioning of the hands. The second method uses different types of cameras. It is based on image processing which does not require equipment such as sensors. This approach only relies on the application of various image processing techniques and pattern recognition. The main purpose of this paper is to recognize and understand hand movements done by those people whose primary language is sign language among other applications that can be developed using this technology as well. This ingenious method goes beyond the confines of traditional gesture identification.

Recognizing ISL (International Sign Language) hand gestures, this also changes these signs into voice thus transcending the visual aspect.

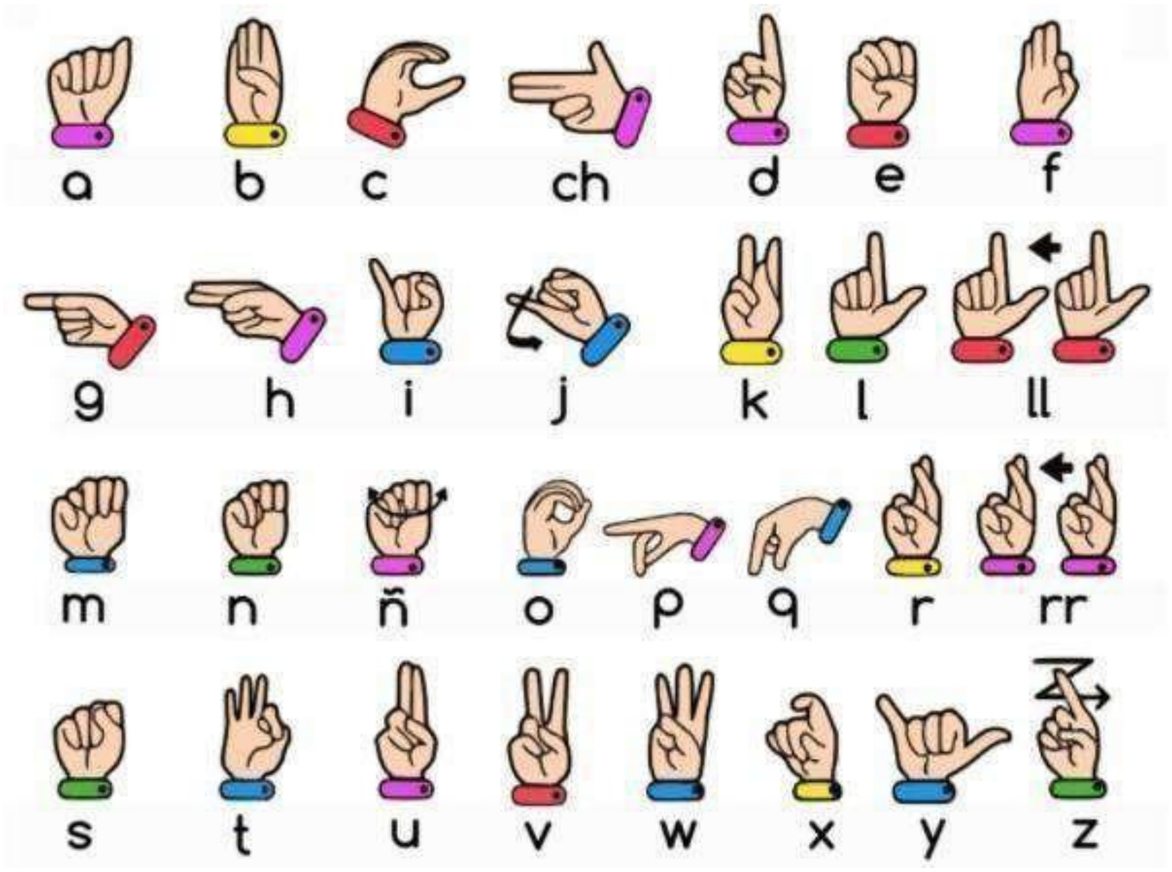


Fig 1.1 Hand Gestures Symbols

1.2 SCOPE OF THE PROJECT

Both those who are deaf or dumb and those who do not comprehend sign language will benefit from this system. All they have to do is use sign language movements, and the system will recognize what they are attempting to communicate. Once identified, it will provide the output in both text and speech format. To develop computer software and train a CNN model that can recognize live International Sign Language hand gestures, display the hand gesture's output in text format, convert it to speech format, and vice versa. Our software utilizes to produce audible voice from this data. With this functionality, our system is much more usable and accessible, which helps people who do not understand sign language as well as those who utilize ISL.

Essentially, this system translates visual ISL movements into comprehensible and accessible written and spoken formats, so establishing a bridge between sign language users and non-users. With this method, we want to promote more inclusion in communication practices by facilitating easy conversation between those who are hard of hearing or deaf and people who are not familiar with sign language. This system aims to open up communication barriers and create an inclusive environment where those who communicate through sign language can communicate with other people using speech or hearing ability without any problems. This system is a major step to ensure that everyone, irrespective of whether they can hear or speak, can communicate with each other. A fascinating division in computer vision technology is the automatic recognition of human gestures from live camera feeds.

CHAPTER 2

LITERATURE SURVEY

Title : Indian Sign language to speech conversion using Convolutional Neural Network.

Authors : Sashidar R,Surendra R Hegde, chinmaya k, Ankit Priyesh, A S Manjunath

Published : IEEE 2022

Description :

True incapacity is the inability to speak. A person who has a speech impediment is unable to communicate with others through speech and hearing. Individuals use sign language as a form of communication to overcome this disability. Even though signing has become commonplace in recent years, it can still be difficult for non-signers to communicate with signers. The flow of information and emotions in a person's life has become increasingly dependent on communication over time. The only way for that person with special needs to communicate with the rest of the world is through sign language, which uses entirely distinct hand motions. With the most recent developments in computer vision and deep learning techniques, there has been significant improvement in the disciplines of motion and gesture identification. For American Sign Language (ASL), sign language recognition has been a well-researched subject. Nevertheless, there aren't much published analysis works on Indian Sign Language (ISL). The intended method will recognise 4972 static hand signs for the twenty-four different English alphabets (A, B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y). The main goal of our effort is to create a deep learning-based application that uses the "Google text to speech" API to translate sign language into text.

Title : Automatic Sign Language Transalation System neural Network Technologies And 3D Animation

Authors : Yevhenii shovkovyi, Olena Grynyova, Serhii udovonko, Larysa Chala, China.

Published : December 2023

Description :

Implementation of automatic sign language translation software in the process of social inclusion of people with hearing impairment is an important task. Social inclusion for people with hearing disabilities is an acute problem that must be solved in the context of the development of IT technologies and legislative initiatives that ensure the rights of people with disabilities and their equal opportunities. This substantiates the relevance of the research of assistive technologies, in the context of software tools, such as the process of social inclusion of people with severe hearing impairment in society. The subject of research is methods of automated sign language translation using intelligent technologies. The purpose of the work is the development and research of sign language automation methods to improve the quality of life of people with hearing impairments in accordance with the "Goals of Sustainable Development of Ukraine" (in the "Reduction of Inequality" part).The main tasks of the research are the development and testing of methods of converting sign language into text, converting text into sign language, as well as automating translation from one sign language to another sign language using modern intelligent technologies. Neural network modeling and 3D animation methods were used to solve these problems.

Title : A Transfer Learning Model for Gesture recognition Based On the Deep Feature Extracted by CNN.

Authors : yongxiang zoo, Long Cheng.

Published : IEEE 2023

Description :

The sEMG-based hand gesture recognition is prevalent in human-computer interface (HCI) systems. However, the generalization of the recognition model does not perform well on cross-subject and cross-day. Transfer learning, which applies the pre-trained model to another task, has demonstrated its effectiveness in solving this kind of problem. In this regard, this paper first proposes a multi-scale kernel convolutional neural network (MKCNN) model to extract and fuse multi-scale features of the multi-channel sEMG signals. Based on the proposed MKCNN model, a transfer learning model named TL-MKCNN combines the MKCNN and its siamese network by a custom distribution normalization module (DNM) and a distribution alignment module (DAM) to realize domain adaptation. The DNM can cluster the deep features extracted from different domains to their category center points embedded in the feature space, and the DAM further aligns the overall distribution of the deep features from different domains. The MKCNN model and the TL-MKCNN model are tested on various benchmark databases to verify the effectiveness of the transfer learning framework.

Title : Sign Language to Text Conversion in Real Time using Transfer Learning.

Authors : Shubham Thakar, Bhavya Shah, Anant Nimkar, Samveg Shah.

Published : Nov 2022

Description :

The people in the world who are hearing impaired face many obstacles in communication and require an interpreter to comprehend what a person is saying. There has been constant scientific research and the existing models lack the ability to make accurate predictions. So we propose a deep learning model trained on the ASL i.e. American Sign Language which will take action in the form of American Sign Language as input and translate it into text. To achieve the former a Convolution Neural Network based VGG16 architecture is used as well as a TensorFlow model for image classification and we have improved the accuracy of the latter by over 4%. There has been an improvement in accuracy from 94% of CNN to 98.7% by Transfer Learning. An application with the deep learning model integrated has also been built..

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Sign All : A leader in assistive technology, SignAll focuses on overcoming communication obstacles that members of the Deaf and Hard of Hearing community encounter. They have created a cutting-edge system that translates sign language into text using computer vision and machine learning.

Real-time translation of the text appears on a screen to facilitate smooth communication between sign language users and non-users. To promote more inclusive communication, this ground-breaking technology has been incorporated into a variety of contexts, such as workplaces, educational institutions, and public service locations.

3.1.1.1 LIMITATIONS OF THE EXISTING SYSTEM

Speech Conversion and Recommendations: The processed text was not converted into voice. The suggestions will not be showed up for the seamless and productive conversation.

Lack of Reverse Process: The Hand gesture is converted into voice and on the other hand, the reverse process is done by converting voice to hand gesture.

3.2 PROPOSED SYSTEM

Our suggested sign language to voice system seeks to improve upon present technology while resolving some of its drawbacks. The idea is to develop a bidirectional, more portable, adaptable, and accurate communication tool that can convert voice into sign language and vice versa. The Algorithm used in this system is CNN - Convolutional Neural Networks are a type of deep learning algorithm primarily used for image processing and recognition tasks. They're particularly well-suited to recognizing patterns in images, making them an excellent choice for sign language detection.

3.2.1 ADVANTAGES OF THE PROPOSED SYSTEM

1. Adequate lighting condition will be enough for the detection and classification of hand gestures.
2. Word Recommendation System.
3. Reverse Process.

3.3 REQUIREMENT SPECIFICATION

3.3.1 HARDWARE REQUIREMENTS

- Computer with Minimum i5 10th Gen Processor
- Camera
- Microphone
- Speaker

3.3.2 SOFTWARE REQUIREMENTS

- Operating System: Windows 8 and Above
- IDE: Visual Studio Code
- Programming Language: Python 3.9 5
- Python libraries: OpenCV, NumPy, Keras, MediaPipes, Tensorflow, Pillow, Speech Recognition, Tkinter

3.4 LANGUAGE SPECIFICATION

- Python : Deep learning frameworks and libraries support Python
- Keras: Version 3.5 – 3.8
- Tensorflow: Version 3.5 – 3.9
- NumPy: Version 1.22.0 – 1.23.5
- OpenCV: 4.7.0
- MediaPipe : Version 0.10.7
- Pillow : Version 9.5.0
- Speech Recognition : 3.10.1
- Tkinter : 8.5

3.5 ALGORITHM DESCRIPTION

CNN is a class of neural networks that are highly useful in solving computer vision problems. They found inspiration from the actual perception of vision that takes place in the visual cortex of our brain. They make use of a filter/kernel to scan through the entire pixel values of the image and make computations by setting appropriate weights to enable detection of a specific feature. CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful tool that can identify features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features.

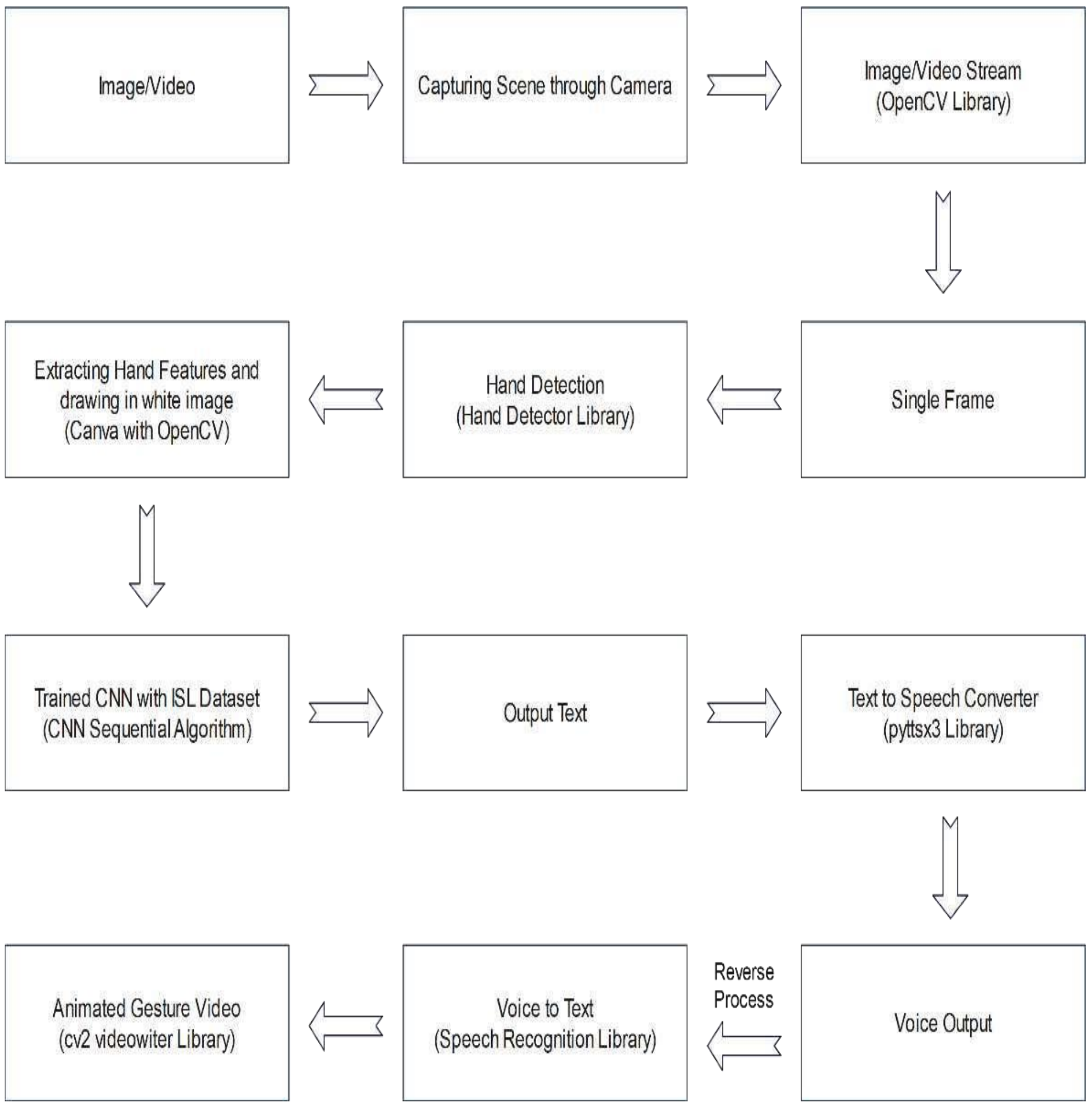
Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions(number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores. Convolution, max pooling, flatten, dense, dropout, and a fully connected neural network layer are among the layers that CNN is composed of. When combined, these layers provide a very potent tool for feature recognition in images. Beginning layers identify low level features, which progressively start to identify higher level features that are more complicated. Rather than being entirely coupled to every other neuron in the layer, a layer's neurons will only be connected to a tiny portion of the layer (window size) preceding it. Furthermore, as the entire imagewill eventually be reduced to a single vector of class scores at the end of the CNN architecture, the final output layer would have dimensions. This exceptional capability stems from the unique architecture of CNNs, featuring specialized layers known as ”convolutional layers”. Imagine these layers as meticulous filters that meticulously scan the input image.

Performing element-wise multiplication with corresponding pixel values. This exceptional capability stems from the unique architecture of CNNs, featuring specialized layers known as ”convolutional layers”. Imagine these layers as meticulous filters that meticulously scan the input image, performing element-wise multiplication with corresponding pixel values. In the realm of computer vision, Convolutional Neural Networks (CNNs) have emerged as a dominant force, revolutionizing the way this system analyzes images and videos. Through this iterative process, CNNs progressively build a hierarchy of knowledge by identifying specific features within the image. Their remarkable ability lies in extracting meaningful visual features, allowing them to ”see” and understand the content of an image in a way that traditional methods often struggle with. Through this iterative process, CNNs progressively build a hierarchy of knowledge by identifying specific features within the image. Initial layers typically focus on detecting fundamental shapes and edges, while later layers learn to recognize more complex patterns and combinations of these features. To address overfitting and manage data size, pooling layers are employed, downsampling the data while retaining crucial information. Finally, fully connected layers process the refined image representations to perform classification or prediction tasks. This distinctive architecture equips CNNs with several noteworthy advantages. One critical benefit lies in automated feature extraction, eliminating the traditionally laborious and time-consuming process of manual feature engineering. Additionally, CNNs exhibit translation invariance, meaning they can identify objects or patterns regardless of their location within the image.

CHAPTER 4

SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM



Image/Video
Capturing Scene through Camera
Image/Video Stream (OpenCV Library)
Single Frame
Hand Detection (Hand Detector Library)
Extracting Hand Features and drawing in white image (Canva with OpenCV)
Trained CNN with ISL Dataset (CNN Sequential Algorithm)
Output Text
Text to Speech Converter (pyttsx3 Library)
Voice Output
Reverse Process
Voice to Text (Speech Recognition Library)
Animated Gesture Video (cv2 videowiter Library)

Fig 4.1. Architecture Diagram

4.2 USECASE DIAGRAM

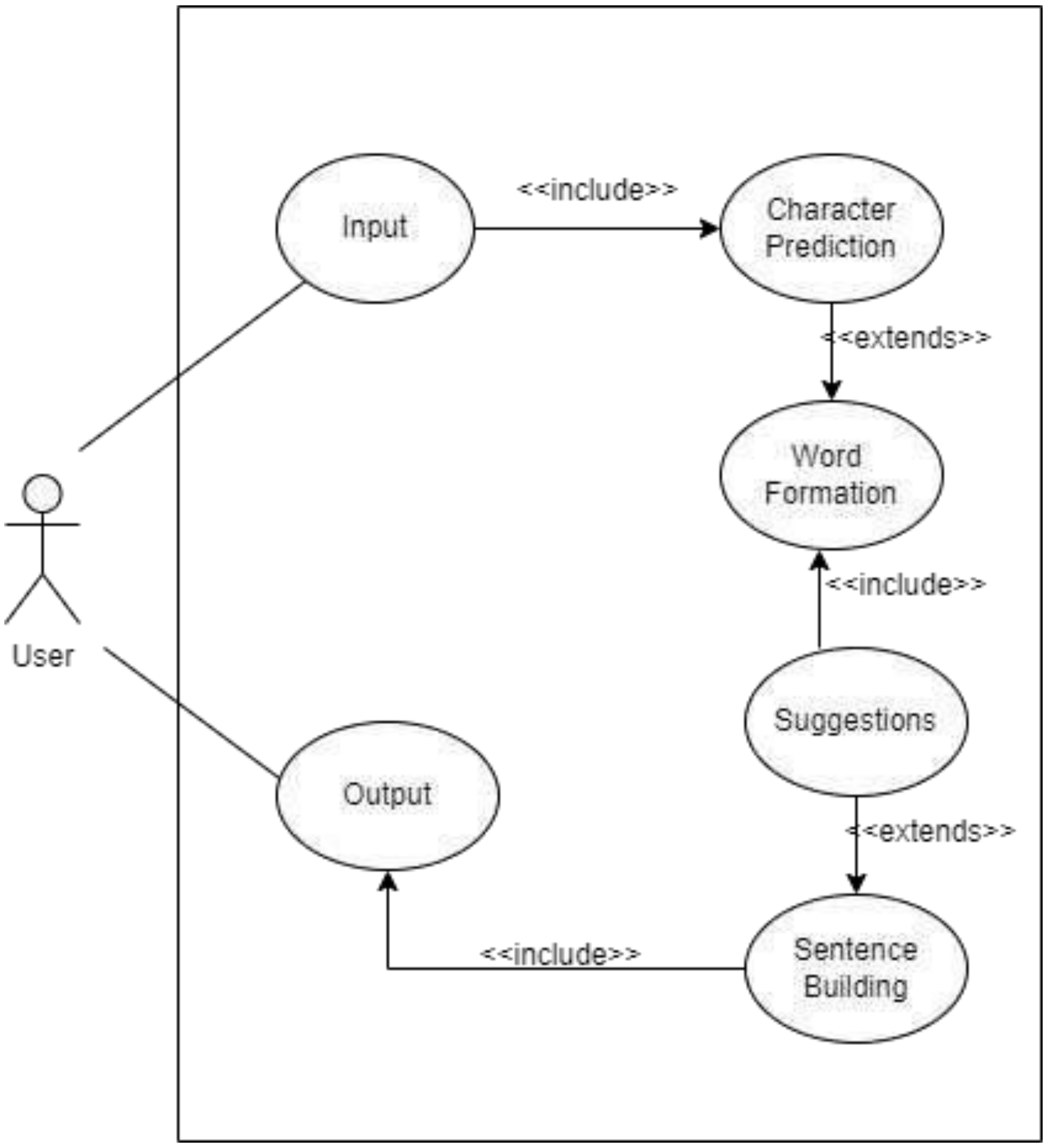


Fig 4.2. Usecase Diagram

4.3 SEQUENCE DIAGRAM

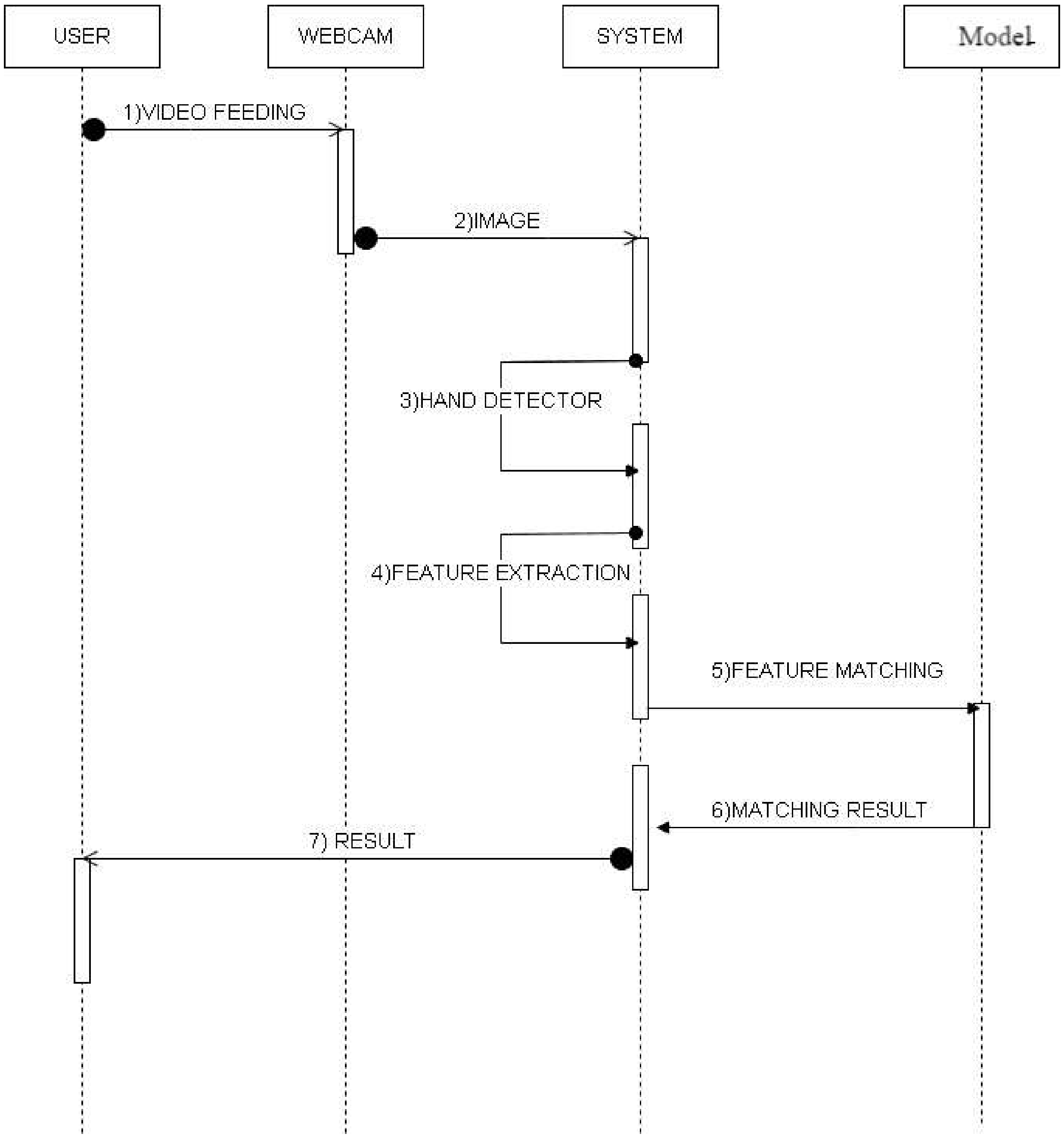


Fig 4.3. Sequential Diagram

4.4 ACTIVITY DIAGRAM

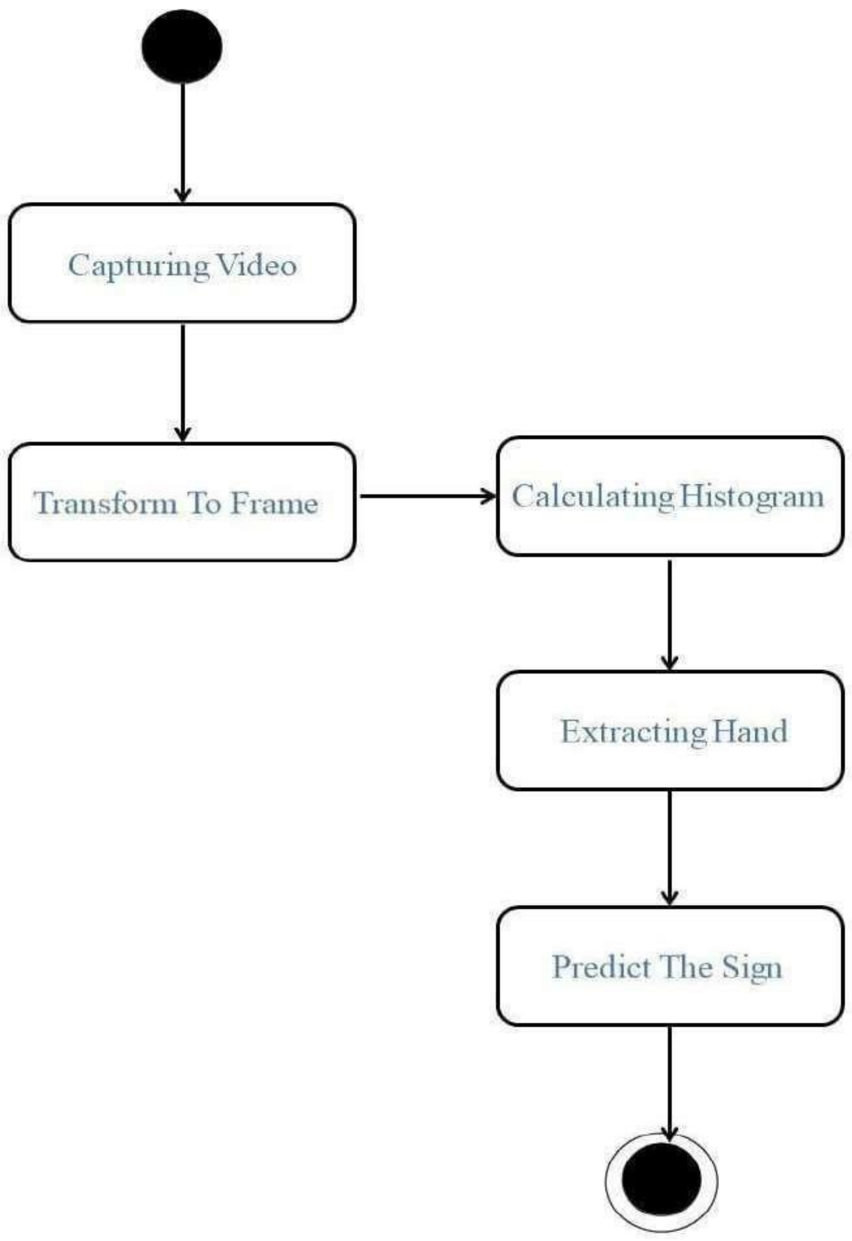


Fig 4.4. Activity Diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES

5.1.1 DATA PROCUREMENT

The different approaches to acquire data about the hand gesture can be done in the Following ways: It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly. In vision-based methods, the computer webcam is the input device for observing the information of hands and/or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices, thereby reducing costs. The main challenge of vision-based hand detection ranges from coping with the large variability of the human hand's appearance due to a huge number of hand movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

5.1.2 DATA NORMALIZATION AND FEATURE SELECTION

Using the HandDetector class from the cvzone module, this module relies heavily on it to find hands in processed frames. The HandDetector applies a complex algorithm that uses aspect ratio as well as filters to improve accuracy in hand detection. On successful detection, the module then proceeds with Region Of Interest (ROI) extraction using the bounding box coordinates of the detected hand, which is essential for isolating the hand area for further processing. Also, making hand landmarks visible on a white background canvas at runtime is significant. This canvas serves as a graphical space for rendering hand landmarks, resulting in a visual representation of the detected hand gesture.

These 21 identified landmarks take each shape of 21 features and lines joining them provide an intricate picture of how hands are placed and move. Accuracy in recognizing hand gestures is improved by integrating aspect ratio and additional filters during hand detection. The program generates a whole image of the 21 hand landmarks on canvas, making up a detailed representation of the structure and pose of the hand. This detailed representation along with theaspect ratio and filters makes sure that presented detected hand gestures are accurate enough for further analysis and interpretation of gestures is achieved.

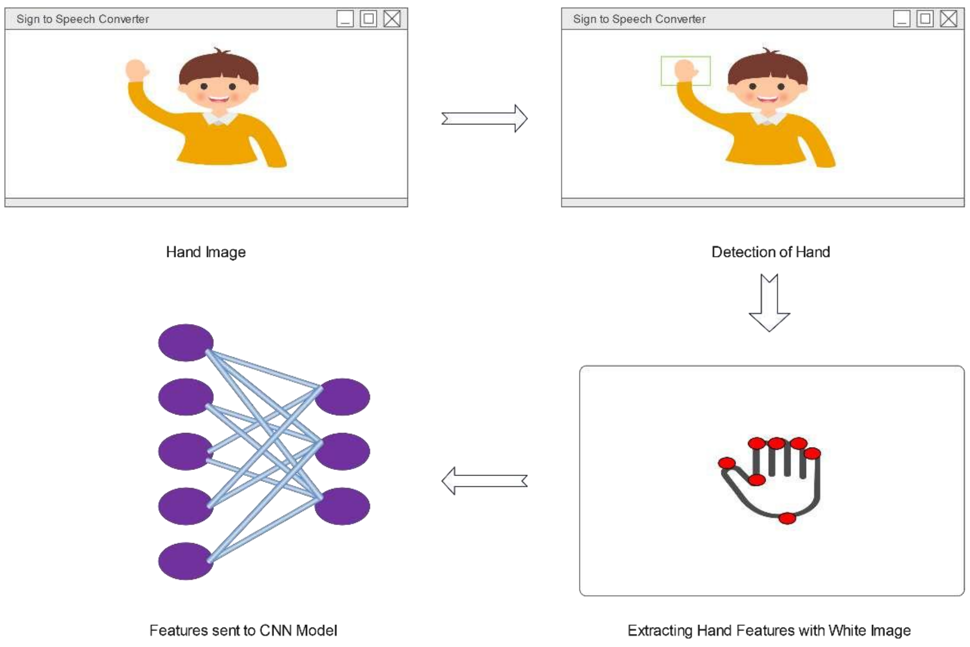


Fig 5.1. Gesture Classification

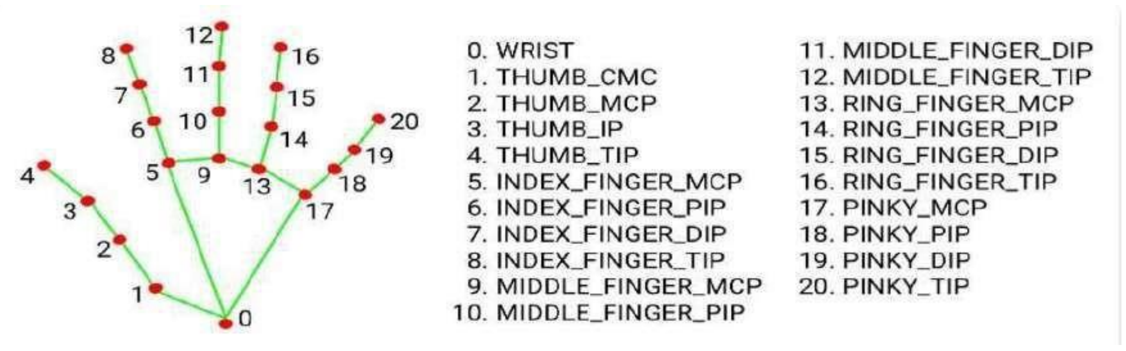


Fig 5.2. Hand Landmark System

5.1.3 GESTURE INTERPRETATION

Layer: I've chosen a little window size for the convolution layer that extends to the input matrix's depth (usually 5 by 5). The layer is made up of window-sized learnable filters. In each iteration, I compute the dot product of theinput values at a particular place and slid the window by the stride size, which isusually 1. As I proceed with this procedure, I will produce a 2- Dimensional activation matrix that displays the matrix's reaction at each spatial location. In other words, the network will pick up filters that turn on when it detects certain kinds of visual features, such a splotch of a particular color.

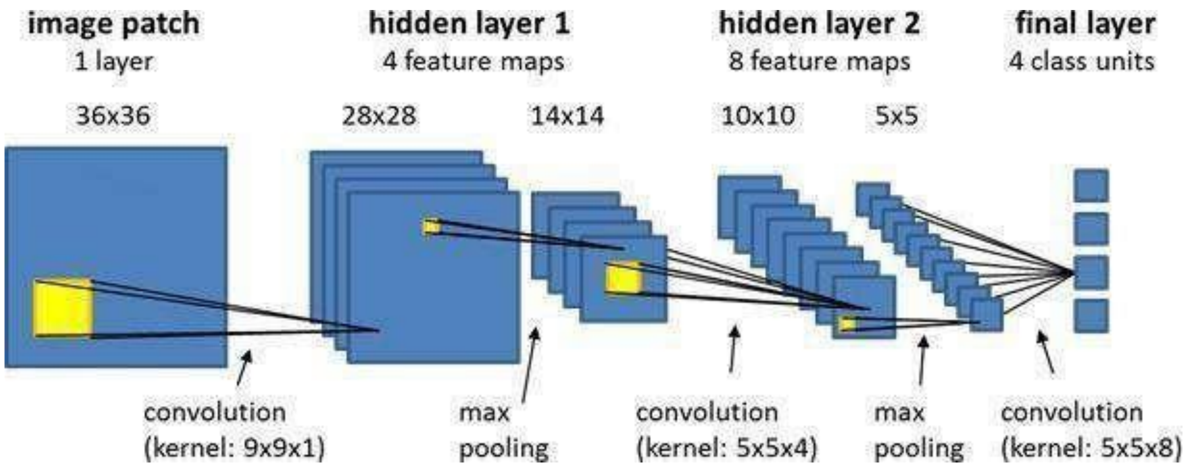


Fig 5.3. CNN Architecture

Pooling Layer: In order to lower the size of the activation matrix and, eventually, the learnable parameters, we employ a pooling layer.

Two categories of pooling exist:

Max Pooling : When using max pooling, we take a window size—for instance, a 2*2 window—and only the highest four values. We'll close this window and carry on until we eventually get an activation matrix that is half the size it was originally.

Average Pooling: We take the average of every value in a window when we use average pooling.

Fully Connected Layer: In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

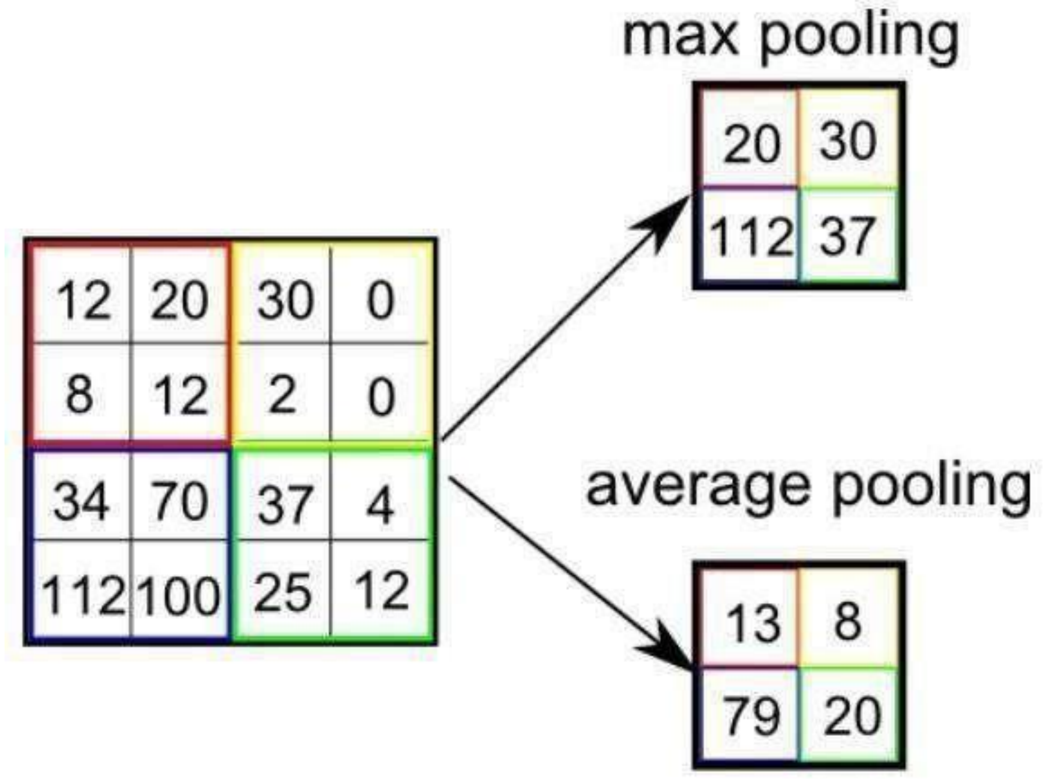


Fig 5.4. Pooling Layer

The preprocessed 180 images/alphabet will feed the keras CNN model. Because we got bad accuracy in 26 different classes thus, We divided whole 26 different alphabets into 8 classes in which every class contains similar alphabets: [y,j], [c,o], [g,h], [b,d,f,I,u,v,k,r,w], [p,q,z], [a,e,m,n,s,t]. All the gesture labels will be assigned with a probability. The label with the highest probability will treated to be the predicted label. So when model will classify [aemnst] in one single class using mathematical operation on hand landmarks we will classify further into single alphabet a or e or m or n or s or t. Finally, we got 97.85% Accuracy (with and without clean background and proper lightning conditions).

method. A proposed neural network architecture that is based on a well - designed Convolutional Neural Network (CNN) specifically optimized for image classification tasks. The initial convolutional layer has 32 filters with a 3x3 kernel size and generates feature maps. By using 32 filters in this way, followed by max- pooling and generating feature maps, the next one in this sequence also involvestwo other convolutional layers that use 16 filters each before additional max- pooling. The last dense layer provides an output prediction where each node corresponds to a class in the classification task. This subpart helps the model to differentiate between nearly similar alphabets.

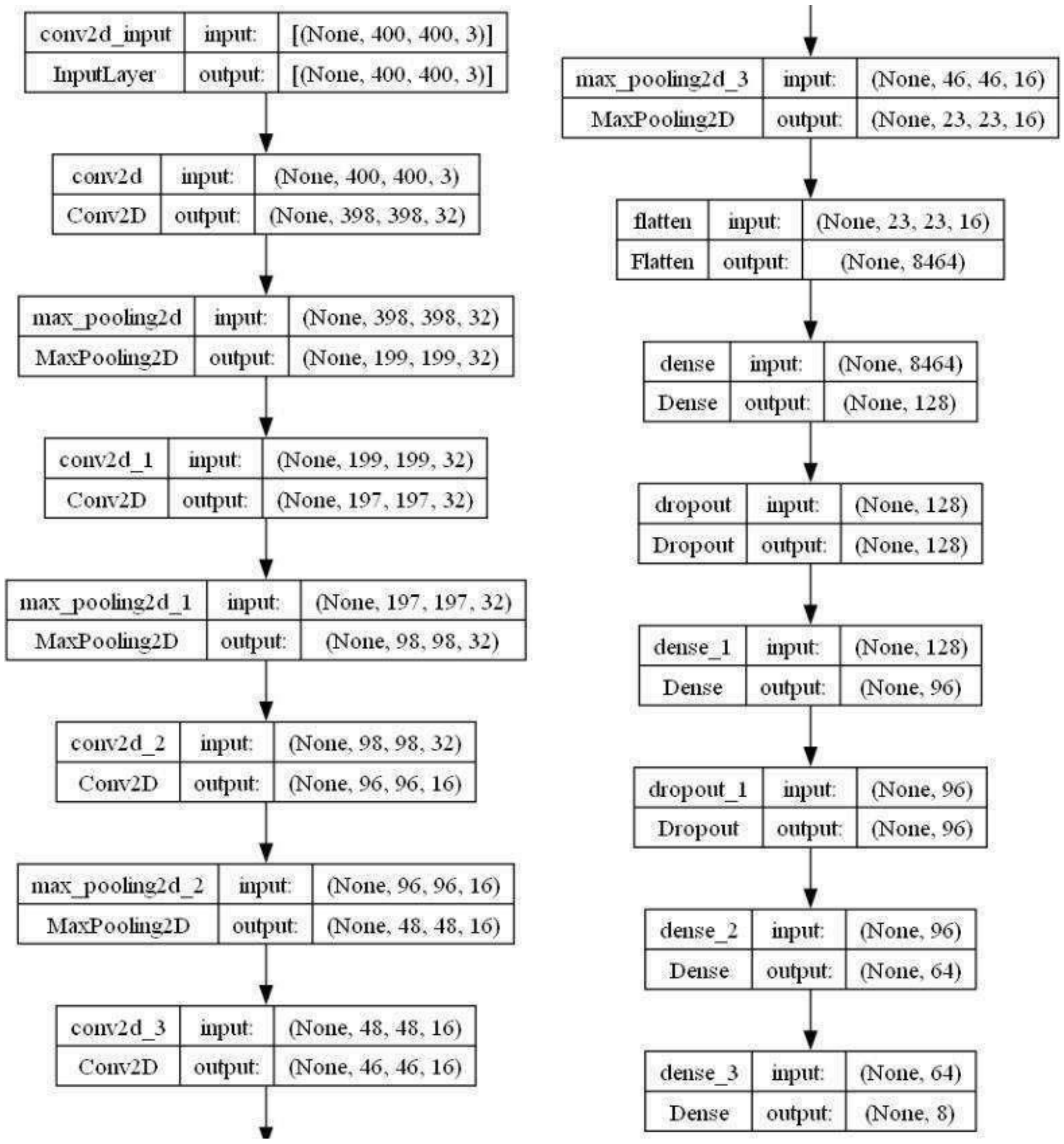

conv2d_input | input: | [(None, 400, 400, 3)]
InputLayer | output: | [(None, 400, 400, 3)]
conv2d | input: | (None, 400, 400, 3)
Conv2D | output: | (None, 398, 398, 32)
max_pooling2d | input: | (None, 398, 398, 32)
MaxPooling2D | output: | (None, 199, 199, 32)
conv2d_1 | input: | (None, 199, 199, 32)
Conv2D | output: | (None, 197, 197, 32)
max_pooling2d_1 | input: | (None, 197, 197, 32)
MaxPooling2D | output: | (None, 98, 98, 32)
conv2d_2 | input: | (None, 98, 98, 32)
Conv2D | output: | (None, 96, 96, 16)
max_pooling2d_2 | input: | (None, 96, 96, 16)
MaxPooling2D | output: | (None, 48, 48, 16)
conv2d_3 | input: | (None, 48, 48, 16)
Conv2D | output: | (None, 46, 46, 16)
max_pooling2d_3 | input: | (None, 46, 46, 16)
MaxPooling2D | output: | (None, 23, 23, 16)
flatten | input: | (None, 23, 23, 16)
Flatten | output: | (None, 8464)
dense | input: | (None, 8464)
Dense | output: | (None, 128)
dropout | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 96)
dropout_1 | input: | (None, 96)
Dropout | output: | (None, 96)
dense_2 | input: | (None, 96)
Dense | output: | (None, 64)
dense_3 | input: | (None, 64)
Dense | output: | (None, 8)

Fig 5.5. The Proposed CNN Model

5.1.4 SPEECH SYNTHESIS

The model translates known gestures into words. we have used pyttsx3 library toconvert the recognized words into the appropriate speech. The text-to-speech output is a simple workaround, but it's a useful feature because it simulates a real-life dialogue. This groundbreaking feature empowers individuals who use sign language to not only communicate effectively with others but also to bridge the communication gap with those unfamiliar with sign language (like ISL). This functionality fosters inclusivity in various settings, such as education, healthcare, and social interactions, by enabling seamless communication between individuals regardless of their language background.

5.1.5 REVERSE PROCESS

The code converts voice input to text using speech_recognition, then processes the text to generate hand gesture animations via OpenCV. Users initiate voice input, observe animated gestures, creating a seamless interaction bridging voice commands and visual representation. The generated animations provide visual feedback, enhancing user understanding. Additionally, error handling ensures robustness, gracefully managing potential issues during voice input and video generation processes, ensuring a smooth user experience. This fusion of voice and visual elements facilitates a more intuitive and engaging user experience. Users can effortlessly initiate voice inputs, witnessing corresponding hand gestures dynamically generated on-screen. Such visual feedback not only enhances comprehension but also augments user engagement and satisfaction. Furthermore, the code's robust error-handling mechanisms ensure uninterrupted functionality, effectively managing potential issues that may arise during voice input processing or animation generation. This commitment to robustness underscores the dedication to delivering a seamless user experience, regardless of potential technical challenges.

CHAPTER 6

CONCLUSION FUTURE AND ENHANCEMENTS

This project introduces an innovative real-time International Sign Language (ISL) gesture recognition and translation system that uses Convolutional Neural Networks (CNNs). The system's excellence is in its ability to recognize ISL gestures with high accuracy 97.85%, making it indeed a reliable tool for people who rely on sign language. By transforming the recognized ISL gestures into audible voice, the technology transcends the visual medium and bridges communication gaps between deaf persons and those using spoken language. The point about this technology is that it can potentially create an all-inclusive and communicative world where language would be borderless. The model was trained achieving a training accuracy of 94.10% and a validation accuracy of 99.25%. The corresponding training loss was 0.1834 while the validation loss was 0.0297. Therefore, these results demonstrate that the model effectively learned underlying patterns in the training data as well as showed high generalization performance on the validation set by utilizing CNN layers having a 3X3 Kernel Size for filtering. It must undergo rigorous testing and refinement based on user feedback to ensure usability and effectiveness. Rigorous testing and refinement driven by user feedback guarantee the usability and efficacy of its use. The adaptability plus robustness of this system makes it applicable in real-world scenarios. This shows that this system can be used in real-life scenarios since it is adaptable as well as robust enough. The future holds great promise in the areas of sign language identification and translation. The research provides a foundation for further investigations into expanding vocabulary, identifying more sign languages and dialects, and improving flexibility in various real-life situations. Such technology will continue to evolve through constant collaboration with the ISL community as well as user testing. This study can be improved through the expansion of the dataset.

include a more comprehensive array of International Sign Language (ISL) letters and words. By augmenting the dataset, there is a potential to enhance the accuracy of the recognition system while simultaneously reducing loss. Furthermore, incorporating additional words and phrases may strengthen the system's capability to predict complete expressions. Subsequently, the integration of a text - to- speech engine could facilitate the conversion of predicted expressions into audible speech. In future research, efforts may be directed toward developing methods to translate ISL gestures into commands understandable by robots or machines. This advancement would enable individuals to interact with robots using ISL gestures as commands, thereby contributing to enhanced accessibility and usability in human-robot interaction scenarios.

The Future enhancements will be the voice is available in all the languages not only in English. Furthermore, the implementation of 2D or 3D Avatar in reverse process. Lastly, the action gestures like Hello, thank you will be recognized by the gesture .

APPENDIX I – SAMPLE CODE

```
import numpy as np
import math
import cv2
import os, sys
import traceback
import pyttsx3
from keras.models import load_model
from cvzone.HandTrackingModule import HandDetector
from string import ascii_uppercase
import enchant
ddd=enchant.Dict("en_US")
hd = HandDetector(maxHands=1)
hd2 = HandDetector(maxHands=1)
import tkinter as tk
from PIL import Image, ImageTk
offset=29
os.environ["THEANO_FLAGS"] = "device=cuda, assert_no_cpu_op=True"
class Application:

    def _init_(self):
        self.vs = cv2.VideoCapture(0)
        self.current_image = None
        self.model = load_model(r"C:\\Users\Naveen\Downloads\Sign-Language-To-
Text-and-Speech-Conversion-master\Sign-Language-To-Text-and-Speech-
Conversion-master\cnn8grps_rad1_model.h5")
        self.speak_engine=pyttsx3.init()
        self.speak_engine.setProperty("rate",100)
        voices=self.speak_engine.getProperty("voices")
        self.speak_engine.setProperty("voice",voices[0].id)

        self.ct = {}
        self.ct['blank'] = 0
        self.blank_flag = 0
        self.space_flag=False
        self.next_flag=True
        self.prev_char=""
        self.count=-1
        self.ten_prev_char=[]
        for i in range(10):
            self.ten_prev_char.append(" ")
```

```
for i in ascii_uppercase:
    self.ct[i] = 0

print("Loaded model from disk")

self.root = tk.Tk()
self.root.title(" our final year project")
self.root.protocol('WM_DELETE_WINDOW', self.destructor)
self.root.geometry("1300x700")

self.panel = tk.Label(self.root)
self.panel.place(x=100, y=3, width=480, height=640)

self.panel2 = tk.Label(self.root)  # initialize image panel
self.panel2.place(x=700, y=115, width=400, height=400)

self.T = tk.Label(self.root)
self.T.place(x=60, y=5)
self.T.config(text="Dai naveen kai katura", font=("Courier", 30, "bold"))

self.panel3 = tk.Label(self.root)  # Current Symbol
self.panel3.place(x=280, y=585)

self.T1 = tk.Label(self.root)
self.T1.place(x=10, y=580)
self.T1.config(text="Character :", font=("Courier", 30, "bold"))

self.panel5 = tk.Label(self.root)  # Sentence
self.panel5.place(x=260, y=632)

self.T3 = tk.Label(self.root)
self.T3.place(x=10, y=632)
self.T3.config(text="Sentence :", font=("Courier", 30, "bold"))

self.T4 = tk.Label(self.root)
self.T4.place(x=10, y=700)
self.T4.config(text="Suggestions :", fg="red", font=("Courier", 30, "bold"))


self.b1=tk.Button(self.root)
self.b1.place(x=390,y=700)

self.b2 = tk.Button(self.root)
self.b2.place(x=590,  y=700)
```

```
        self.b3 = tk.Button(self.root)
        self.b3.place(x=790, y=700)

        self.b4 = tk.Button(self.root)
        self.b4.place(x=990, y=700)

        self.speak = tk.Button(self.root)
        self.speak.place(x=1305, y=680)
        self.speak.config(text="Speak", font=("Courier", 20), wraplength=100,
command=self.speak_fun)

        self.clear = tk.Button(self.root)
        self.clear.place(x=1205, y=630)
        self.clear.config(text="Clear", font=("Courier", 20), wraplength=100,
command=self.clear_fun)

        self.str = " "
        self.ccc=0
        self.word = " "
        self.current_symbol = "C"
        self.photo = "Empty"


        self.word1=" "
        self.word2 = " "
        self.word3 = " "
        self.word4 = " "

        self.video_loop()

    def video_loop(self):
        try:
            ok, frame = self.vs.read()
            cv2image = cv2.flip(frame, 1)
            hands = hd.findHands(cv2image, draw=False, flipType=True)
            cv2image_copy=np.array(cv2image)
            cv2image = cv2.cvtColor(cv2image, cv2.COLOR_BGR2RGB)
            self.current_image = Image.fromarray(cv2image)
            imgtk = ImageTk.PhotoImage(image=self.current_image)
            self.panel.imgtk = imgtk
            self.panel.config(image=imgtk)

            if hands:
```

```
            # #print(" ---------- lmlist=",hands[1])
            hands= hands[0]

            x, y, w, h = hand['bbox']
            image = cv2image_copy[y - offset:y + h + offset, x - offset:x + w + offset]

            white = cv2.imread("C:\\Users\\devansh
raval\\PycharmProjects\\pythonProject\\white.jpg")
            # img_final=img_final1=img_final2=0

            handz = hd2.findHands(image, draw=False, flipType=True)
            print(" ", self.ccc)
            self.ccc += 1
            if handz:
                hand = handz[0]
                self.pts = hand['lmList']
                # x1,y1,w1,h1=hand['bbox']

                os = ((400 - w) // 2) - 15
                os1 = ((400 - h) // 2) - 15
                for t in range(0, 4, 1):
                    cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                             (0, 255, 0), 3)
                for t in range(5, 8, 1):
                    cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                             (0, 255, 0), 3)
                for t in range(9, 12, 1):
                    cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                             (0, 255, 0), 3)
                for t in range(13, 16, 1):
                    cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                             (0, 255, 0), 3)
                for t in range(17, 20, 1):
                    cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                             (0, 255, 0), 3)
                cv2.line(white, (self.pts[5][0] + os, self.pts[5][1] + os1), (self.pts[9][0] +
os, self.pts[9][1] + os1), (0, 255, 0),
                         3)
                cv2.line(white, (self.pts[9][0] + os, self.pts[9][1] + os1), (self.pts[13][0] +
os, self.pts[13][1] + os1), (0, 255, 0),
                    3)
                cv2.line(white, (self.pts[13][0] + os, self.pts[13][1] + os1), (self.pts[17][0]
+ os, self.pts[17][1] + os1),
                    (0, 255, 0), 3)
                cv2.line(white, (self.pts[0][0] + os, self.pts[0][1] + os1), (self.pts[5][0] +
os, self.pts[5][1] + os1), (0, 255, 0),
                    3)
                cv2.line(white, (self.pts[0][0] + os, self.pts[0][1] + os1), (self.pts[17][0] +
os, self.pts[17][1] + os1), (0, 255, 0),
                    3)

                for i in range(21):
                    cv2.circle(white, (self.pts[i][0] + os, self.pts[i][1] + os1), 2, (0, 0, 255),
1)

                res=white
                self.predict(res)

                self.current_image2 = Image.fromarray(res)

                imgtk = ImageTk.PhotoImage(image=self.current_image2)

                self.panel2.imgtk = imgtk
                self.panel2.config(image=imgtk)

                self.panel3.config(text=self.current_symbol, font=("Courier", 30))

                self.b1.config(text=self.word1, font=("Courier", 20), wraplength=825,
command=self.action1)
                self.b2.config(text=self.word2, font=("Courier", 20), wraplength=825,
command=self.action2)
                self.b3.config(text=self.word3, font=("Courier", 20), wraplength=825,
command=self.action3)
                self.b4.config(text=self.word4, font=("Courier", 20), wraplength=825,
command=self.action4)

            self.panel5.config(text=self.str, font=("Courier", 30), wraplength=1025)
        except Exception:
            print("==", traceback.format_exc())
        finally:
            self.root.after(1, self.video_loop)

    def distance(self,x,y):
        return math.sqrt(((x[0] - y[0]) ** 2) + ((x[1] - y[1]) ** 2))

    def action1(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str = self.str[:idx_word]
        self.str = self.str + self.word1.upper()



    def action2(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str=self.str[:idx_word]
        self.str=self.str+self.word2.upper()
        #self.str[idx_word:last_idx] = self.word2
    def action3(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str = self.str[:idx_word]
        self.str = self.str + self.word3.upper()
    def action4(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str = self.str[:idx_word]
        self.str = self.str + self.word4.upper()
    def speak_fun(self):
        self.speak_engine.say(self.str)
        self.speak_engine.runAndWait()
    def clear_fun(self):
        self.str=" "
        self.word1 = " "
        self.word2 = " "
        self.word3 = " "
        self.word4 = " "

    def predict(self, test_image):
        white=test_image
        white = white.reshape(1, 400, 400, 3)
        prob = np.array(self.model.predict(white)[0], dtype='float32')
        ch1 = np.argmax(prob, axis=0)
        prob[ch1] = 0
        ch2 = np.argmax(prob, axis=0)
        prob[ch2] = 0
        ch3 = np.argmax(prob, axis=0)
        prob[ch3] = 0

        pl = [ch1, ch2]

        # condition for [Aemnst]
        l = [[5, 2], [5, 3], [3, 5], [3, 6], [3, 0], [3, 2], [6, 4], [6, 1], [6, 2], [6, 6], [6, 7], [6, 
0], [6, 5],
             [4, 1], [1, 0], [1, 1], [6, 3], [1, 6], [5, 6], [5, 1], [4, 5], [1, 4], [1, 5], [2, 0], [2, 
6], [4, 6],
             [1, 0], [5, 7], [1, 6], [6, 1], [7, 6], [2, 5], [7, 1], [5, 4], [7, 0], [7, 5], [7, 2]]
        if pl in l:
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and 
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]):
                ch1 = 0
                # print("00000")

        # condition for [o][s]
        l = [[2, 2], [2, 1]]
        if pl in l:
            if (self.pts[5][0] < self.pts[4][0]):
                ch1 = 0
                print("++++++++++++++++++")
                # print("00000")

        # condition for [c0][aemnst]
        l = [[0, 0], [0, 6], [0, 2], [0, 5], [0, 1], [0, 7], [5, 2], [7, 6], [7, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[4][0] and 
self.pts[0][0] > self.pts[12][0] and self.pts[0][0] > self.pts[16][
                0] and self.pts[0][0] > self.pts[20][0]) and self.pts[5][0] > self.pts[4][0]:
                ch1 = 2
                # print("22222")

        # condition for [c0][aemnst]
        l = [[6, 0], [6, 6], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[8], self.pts[16]) < 52:
                ch1 = 2
                # print("22222")



        # condition for [gh][bdfikruvw]
        l = [[1, 4], [1, 5], [1, 6], [1, 3], [1, 0]]
        pl = [ch1, ch2]

        if pl in l:
            if self.pts[6][1] > self.pts[8][1] and self.pts[14][1] < self.pts[16][1] and
self.pts[18][1] < self.pts[20][1] and self.pts[0][0] < self.pts[8][
                0] and self.pts[0][0] < self.pts[12][0] and self.pts[0][0] < self.pts[16][0] and
self.pts[0][0] < self.pts[20][0]:
                ch1 = 3
                print("33333c")
# con for [gh][l]
        l = [[4, 6], [4, 1], [4, 5], [4, 3], [4, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][0] > self.pts[0][0]:
                ch1 = 3
                print("33333b")
# con for [gh][pqz]
        l = [[5, 3], [5, 0], [5, 7], [5, 4], [5, 2], [5, 1], [5, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[2][1] + 15 < self.pts[16][1]:
                ch1 = 3
                print("33333a")
 # con for [l][x]
        l = [[6, 4], [6, 1], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[4], self.pts[11]) > 55:
                ch1 = 4
                # print("44444") # con for [l][d]
        l = [[1, 4], [1, 6], [1, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.distance(self.pts[4], self.pts[11]) > 50) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]):
                ch1 = 4
                # print("44444")

        # con for [l][gh]
        l = [[3, 6], [3, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[4][0] < self.pts[0][0]):
                ch1 = 4
                # print("44444")

        # con for [l][c0]
        l = [[2, 2], [2, 5], [2, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[1][0] < self.pts[12][0]):
                ch1 = 4
                # print("44444")

        # con for [l][c0]
        l = [[2, 2], [2, 5], [2, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[1][0] < self.pts[12][0]):
                ch1 = 4
                # print("44444")

        # con for [gh][z]
        l = [[3, 6], [3, 5], [3, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]) and self.pts[4][1] > self.pts[10][1]:
                ch1 = 5
                print("55555b")

        # con for [gh][pq]
        l = [[3, 2], [3, 1], [3, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][1] + 17 > self.pts[8][1] and self.pts[4][1] + 17 > self.pts[12][1]
and self.pts[4][1] + 17 > self.pts[16][1] and self.pts[4][
                1] + 17 > self.pts[20][1]:
                ch1 = 5
            print("55555a")

        # con for [l][pqz]
        l = [[4, 4], [4, 5], [4, 2], [7, 5], [7, 6], [7, 0]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][0] > self.pts[0][0]:
                ch1 = 5
                # print("55555")

        # con for [pqz][aemnst]
        l = [[0, 2], [0, 6], [0, 1], [0, 5], [0, 0], [0, 7], [0, 4], [0, 3], [2, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[0][0] < self.pts[8][0] and self.pts[0][0] < self.pts[12][0] and
self.pts[0][0] < self.pts[16][0] and self.pts[0][0] < self.pts[20][0]:
                ch1 = 5
                # print("55555")

        # con for [pqz][yj]
        l = [[5, 7], [5, 2], [5, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[3][0] < self.pts[0][0]:
                ch1 = 7
                # print("77777")

        # con for [l][yj]
        l = [[4, 6], [4, 2], [4, 4], [4, 1], [4, 5], [4, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[6][1] < self.pts[8][1]:
                ch1 = 7
                # print("77777")

        # con for [x][yj]
        l = [[6, 7], [0, 7], [0, 1], [0, 0], [6, 4], [6, 6], [6, 5], [6, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[18][1] > self.pts[20][1]:
                ch1 = 7
                # print("77777")

        # condition for [x][aemnst]
        l = [[0, 4], [0, 2], [0, 3], [0, 1], [0, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] > self.pts[16][0]:
                ch1 = 6
                print("666661")



        # condition for [yj][x]
        print("2222  ch1=+++++++++++++++++", ch1, ",", ch2)
        l = [[7, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[18][1] < self.pts[20][1] and self.pts[8][1] < self.pts[10][1]:
                ch1 = 6
                print("666662")

        # condition for [c0][x]
        l = [[2, 1], [2, 2], [2, 6], [2, 7], [2, 0]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[8], self.pts[16]) > 50:
                ch1 = 6
                print("666663")
        l = [[4, 6], [4, 2], [4, 1], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[4], self.pts[11]) < 60:
                ch1 = 6
                print("666664")
      l = [[1, 4], [1, 6], [1, 0], [1, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] - self.pts[4][0] - 15 > 0:
                ch1 = 6
                print("666665")
        l = [[5, 0], [5, 1], [5, 4], [5, 5], [5, 6], [6, 1], [7, 6], [0, 2], [7, 1], [7, 4], [6, 6], [7,
2], [5, 0],
             [6, 3], [6, 4], [7, 5], [7, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
                1]):
```

```
            ch1 = 1
            print("111111")

        # con for [f][pqz]
        l = [[6, 1], [6, 0], [0, 3], [6, 4], [2, 2], [0, 6], [6, 2], [7, 6], [4, 6], [4, 1], [4, 2], [0,
2], [7, 1],
             [7, 4], [6, 6], [7, 2], [7, 5], [7, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and
                    self.pts[18][1] > self.pts[20][1]):
                ch1 = 1
                print("111112")

        l = [[6, 1], [6, 0], [4, 2], [4, 1], [4, 6], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[10][1] > self.pts[12][1] and self.pts[14][1] > self.pts[16][1] and
                    self.pts[18][1] > self.pts[20][1]):
                ch1 = 1
                print("111112")
        fg = 19
        # print("_ch1=",ch1," ch2=",ch2)
        l = [[5, 0], [3, 4], [3, 0], [3, 1], [3, 5], [5, 5], [5, 4], [5, 1], [7, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1]) and (self.pts[2][0] < self.pts[0][0]) and
self.pts[4][1] > self.pts[14][1]):
                ch1 = 1
                print("111113")

        l = [[4, 1], [4, 2], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.distance(self.pts[4], self.pts[11]) < 50) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]):
                ch1 = 1
                print("1111993")
        l = [[3, 4], [3, 0], [3, 1], [3, 5], [3, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1]) and (self.pts[2][0] < self.pts[0][0]) and
self.pts[14][1] < self.pts[4][1]):
                ch1 = 1
                print("1111mmm3")

        l = [[6, 6], [6, 4], [6, 1], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] - self.pts[4][0] - 15 < 0:
                ch1 = 1
                print("1111140")

        # con for [i][pqz]
        l = [[5, 4], [5, 5], [5, 1], [0, 3], [0, 7], [5, 0], [0, 2], [6, 2], [7, 5], [7, 1], [7, 6], [7,
7]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] > self.pts[20][1])):
                ch1 = 1
                print("111114")
        l = [[1, 5], [1, 7], [1, 1], [1, 6], [1, 3], [1, 0]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[4][0] < self.pts[5][0] + 15) and (
            (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
             self.pts[18][1] > self.pts[20][1])):
                ch1 = 7
                print("111114lll;;p")
        l = [[5, 5], [5, 0], [5, 4], [5, 1], [4, 6], [4, 1], [7, 6], [3, 0], [3, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1])) and self.pts[4][1] > self.pts[14][1]:
                ch1 = 1
                print("111115")
        fg = 13
        l = [[3, 5], [3, 0], [3, 6], [5, 1], [4, 1], [2, 0], [5, 0], [5, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if not (self.pts[0][0] + fg < self.pts[8][0] and self.pts[0][0] + fg <
self.pts[12][0] and self.pts[0][0] + fg < self.pts[16][0] and
                    self.pts[0][0] + fg < self.pts[20][0]) and not (
                    self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[12][0] and
self.pts[0][0] > self.pts[16][0] and self.pts[0][0] > self.pts[20][
                0]) and self.distance(self.pts[4], self.pts[11]) < 50:
                ch1 = 1
                print("111116")

        l = [[5, 0], [5, 5], [0, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1]:
                ch1 = 1
                print("1117")
        if ch1 == 0:
            ch1 = 'S'
            if self.pts[4][0] < self.pts[6][0] and self.pts[4][0] < self.pts[10][0] and
self.pts[4][0] < self.pts[14][0] and self.pts[4][0] < self.pts[18][0]:
                ch1 = 'A'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] < self.pts[10][0] and
self.pts[4][0] < self.pts[14][0] and self.pts[4][0] < self.pts[18][
                0] and self.pts[4][1] < self.pts[14][1] and self.pts[4][1] < self.pts[18][1]:
                ch1 = 'T'
            if self.pts[4][1] > self.pts[8][1] and self.pts[4][1] > self.pts[12][1] and
self.pts[4][1] > self.pts[16][1] and self.pts[4][1] > self.pts[20][1]:
                ch1 = 'E'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] > self.pts[10][0] and
self.pts[4][0] > self.pts[14][0] and self.pts[4][1] < self.pts[18][1]:
                ch1 = 'M'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] > self.pts[10][0] and
self.pts[4][1] < self.pts[18][1] and self.pts[4][1] < self.pts[14][1]:
                ch1 = 'N'

        if ch1 == 2:
            if self.distance(self.pts[12], self.pts[4]) > 42:
                ch1 = 'C'
            else:
                ch1 = 'O'
        if ch1 == 3:
            if (self.distance(self.pts[8], self.pts[12])) > 72:
                ch1 = 'G'
            else:
                ch1 = 'H'

        if ch1 == 7:
            if self.distance(self.pts[8], self.pts[4]) > 42:
                ch1 = 'Y'
            else:
                ch1 = 'J'

        if ch1 == 4:
            ch1 = 'L'

        if ch1 == 6:
            ch1 = 'X'

        if ch1 == 5:
            if self.pts[4][0] > self.pts[12][0] and self.pts[4][0] > self.pts[16][0] and
    self.pts[4][0] > self.pts[20][0]:
                if self.pts[8][1] < self.pts[5][1]:
                    ch1 = 'Z'
                else:
                    ch1 = 'Q'
            else:
                ch1 = 'P'

        if ch1 == 1:
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
                1]):
                ch1 = 'B'
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]):
                ch1 = 'D'
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
                1]):
                ch1 = 'F'
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] > self.pts[20][
                1]):
                ch1 = 'I'
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] > self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]):
                ch1 = 'W'
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]) and self.pts[4][1] < self.pts[9][1]:
                ch1 = 'K'
            if ((self.distance(self.pts[8], self.pts[12]) - self.distance(self.pts[6],
    self.pts[10])) < 8) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]):
                ch1 = 'U'
            if ((self.distance(self.pts[8], self.pts[12]) - self.distance(self.pts[6],
    self.pts[10])) >= 8) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]) and (self.pts[4][1] > self.pts[9][1]):
                ch1 = 'V'

            if (self.pts[8][0] > self.pts[12][0]) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]):
                ch1 = 'R'

        if ch1 == 1 or ch1 =='E' or ch1 =='S' or ch1 =='X' or ch1 =='Y' or ch1 =='B':
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
    self.pts[14][1] < self.pts[16][1] and self.pts[18][1] > self.pts[20][1]):
                ch1=" "




        print(self.pts[4][0] < self.pts[5][0])
        if ch1 == 'E' or ch1=='Y' or ch1=='B':
            if (self.pts[4][0] < self.pts[5][0]) and (self.pts[6][1] > self.pts[8][1] and
    self.pts[10][1] > self.pts[12][1] and self.pts[14][1] > self.pts[16][1] and self.pts[18][1]
    > self.pts[20][1]):
                ch1="next"
```

```
if ch1 == 'Next' or 'B' or 'C' or 'H' or 'F' or 'X':
    if (self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[12][0] and
self.pts[0][0] > self.pts[16][0] and self.pts[0][0] > self.pts[20][0]) and (self.pts[4][1] <
self.pts[8][1] and self.pts[4][1] < self.pts[12][1] and self.pts[4][1] < self.pts[16][1] and
self.pts[4][1] < self.pts[20][1]) and (self.pts[4][1] < self.pts[6][1] and self.pts[4][1] <
self.pts[10][1] and self.pts[4][1] < self.pts[14][1] and self.pts[4][1] < self.pts[18][1]):
        ch1 = 'Backspace'


if ch1=="next" and self.prev_char!="next":
    if self.ten_prev_char[(self.count-2)%10]!="next":
        if self.ten_prev_char[(self.count-2)%10]=="Backspace":
            self.str=self.str[0:-1]
        else:
            if self.ten_prev_char[(self.count - 2) % 10] != "Backspace":
                self.str = self.str + self.ten_prev_char[(self.count-2)%10]
    else:
        if self.ten_prev_char[(self.count - 0) % 10] != "Backspace":
            self.str = self.str + self.ten_prev_char[(self.count - 0) % 10]


if ch1=="  " and self.prev_char!="  ":
    self.str = self.str + "  "

self.prev_char=ch1
self.current_symbol=ch1
self.count += 1
self.ten_prev_char[self.count%10]=ch1


if len(self.str.strip())!=0:
    st=self.str.rfind(" ")
    ed=len(self.str)
    word=self.str[st+1:ed]
    self.word=word
    print("---------- word = ",word)
    if len(word.strip())!=0:
        ddd.check(word)
        lenn = len(ddd.suggest(word))
        if lenn >= 4:
            self.word4 = ddd.suggest(word)[3]

        if lenn >= 3:
            self.word3 = ddd.suggest(word)[2]
        if lenn >= 2:
            self.word2 = ddd.suggest(word)[1]

        if lenn >= 1:
            self.word1 = ddd.suggest(word)[0]
    else:
        self.word1 = " "
        self.word2 = " "
        self.word3 = " "
        self.word4 = " "
def destructor(self):

print("Closing Application...")
print(self.ten_prev_char)
self.root.destroy()
self.vs.release()
cv2.destroyAllWindows()
print("Starting Application...")

(Application()).root.mainloop()
```

APPENDIX II – OUTPUT SCREENSHOTS

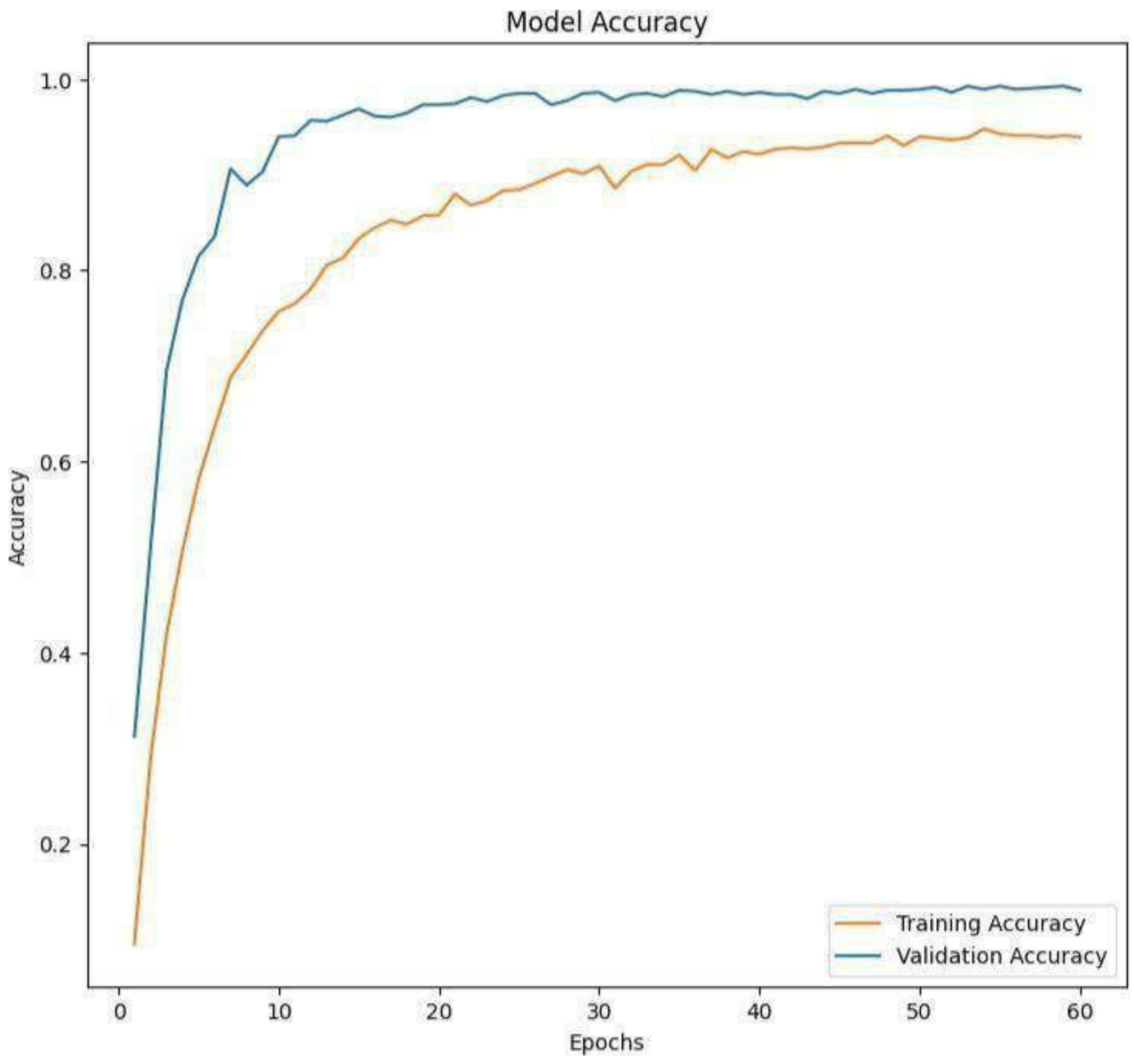


Fig 6.1. Training and Validation Accuracy

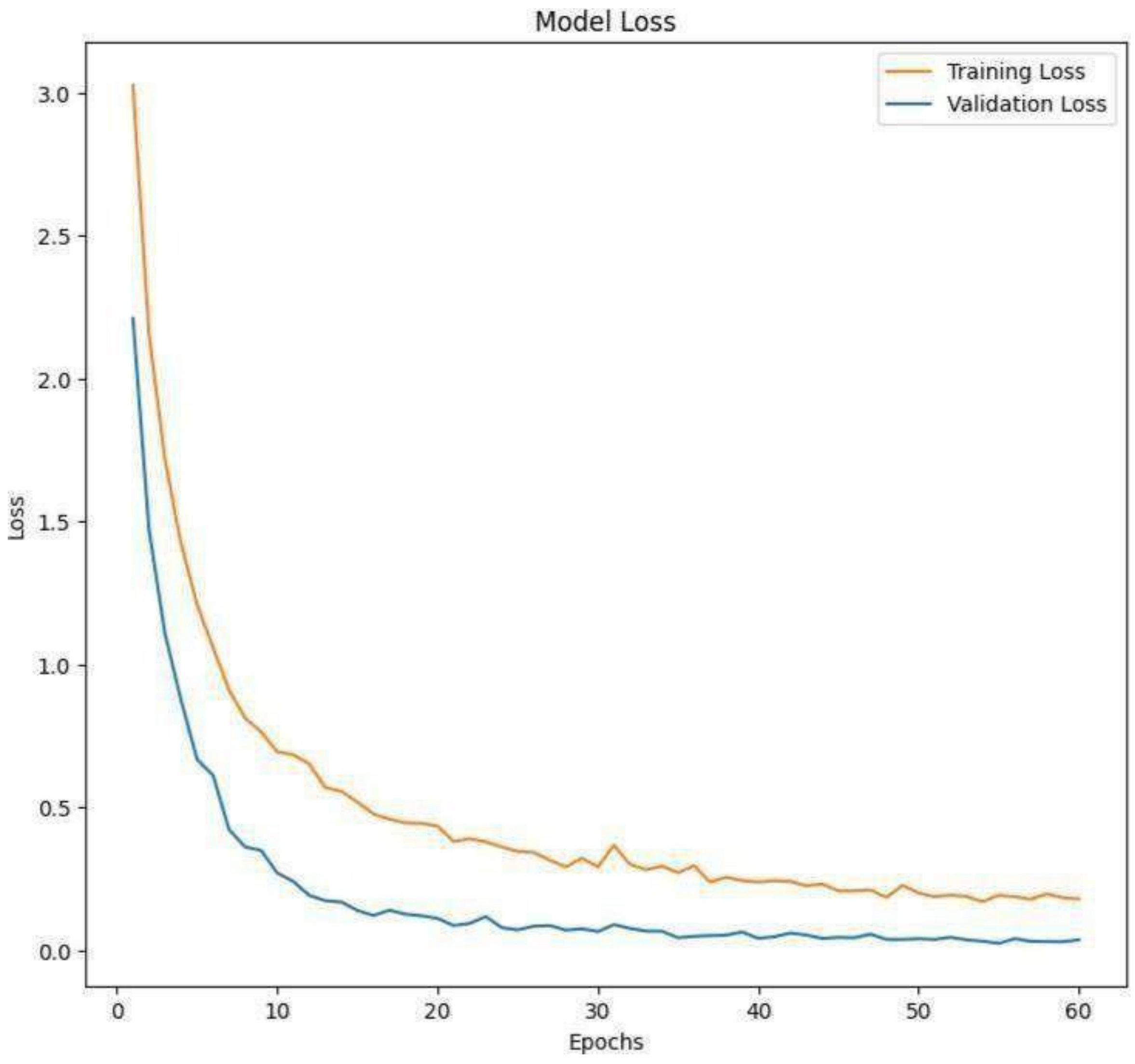


Fig 6.2. Training and Validation Loss

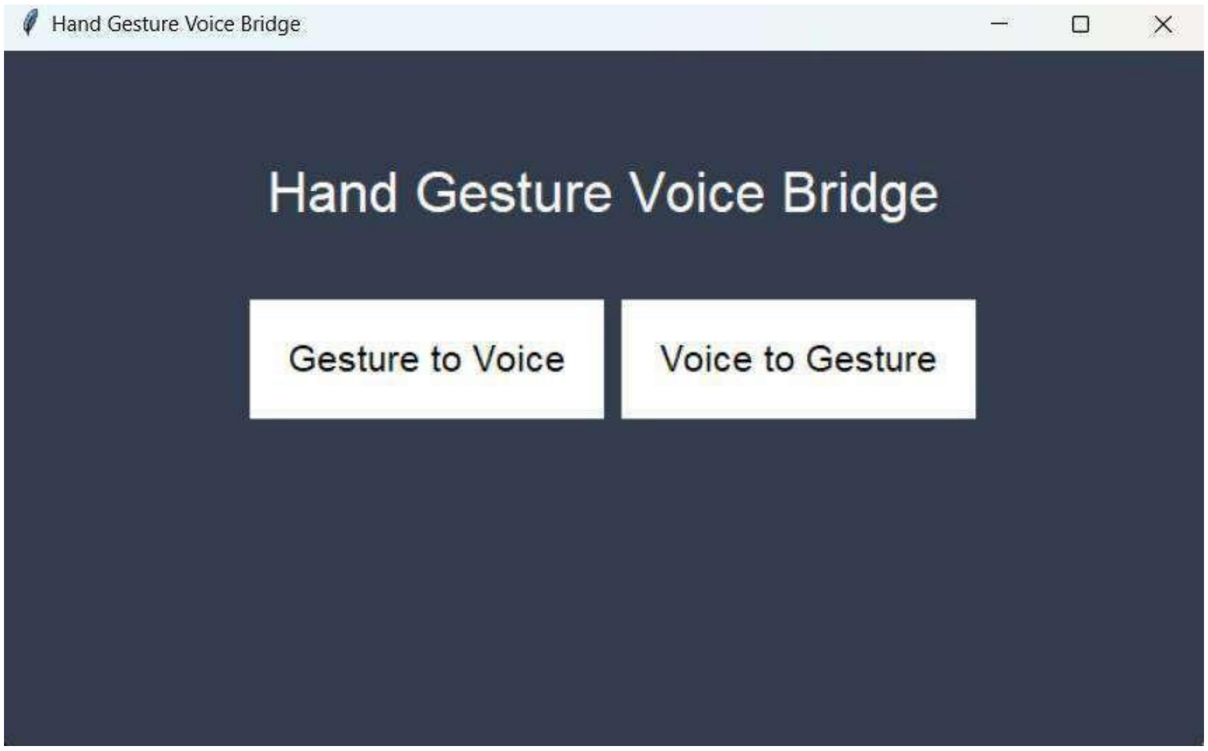
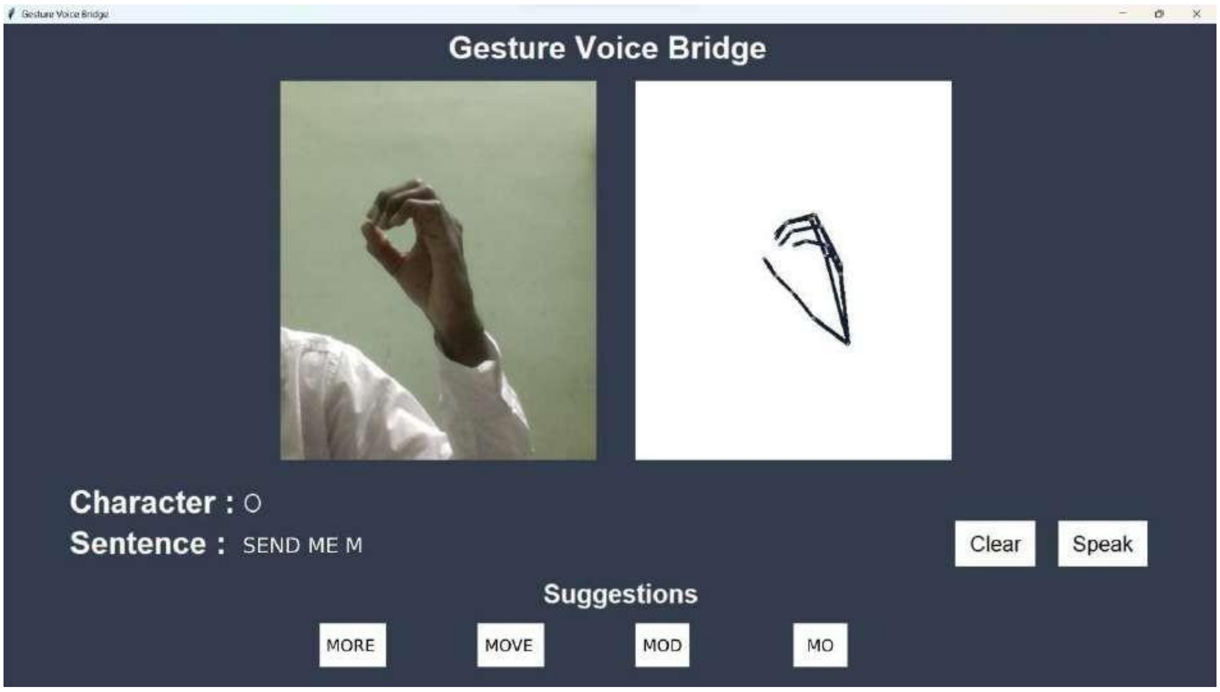


Fig 6.3. Initial Screen



Character : O
Sentence : SEND ME M

Fig 6.4. Gesture to Voice Conversion Module

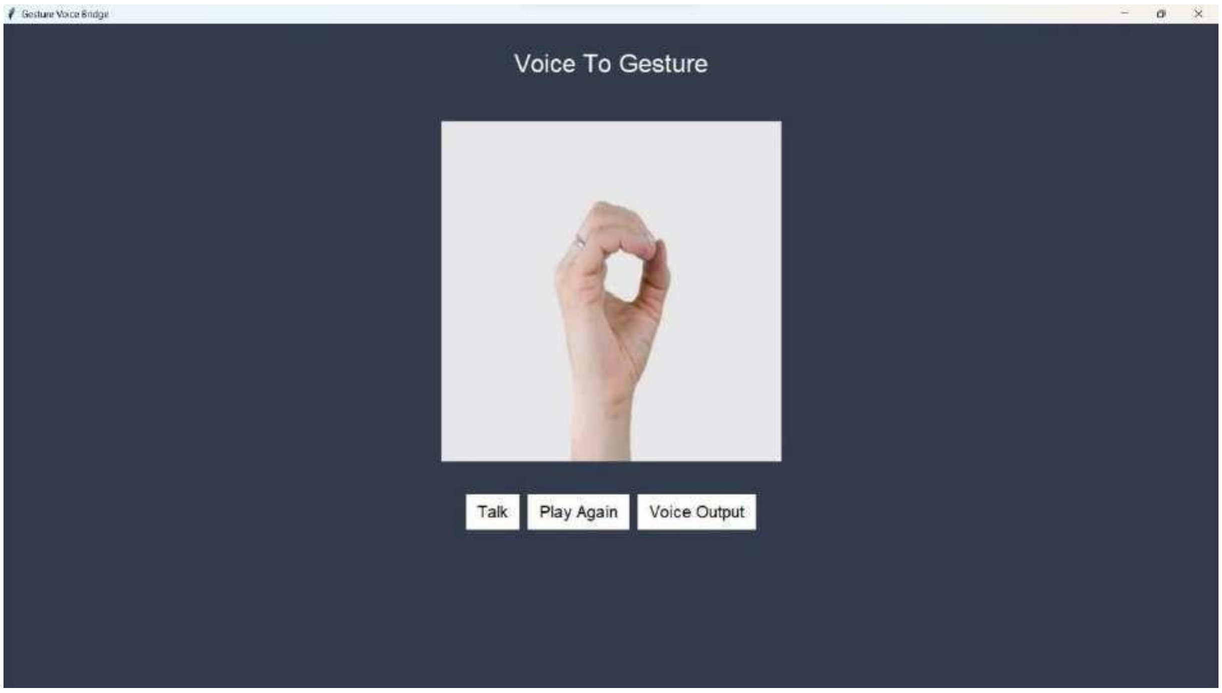


Fig 6.5. Voice to Gesture Conversion Module

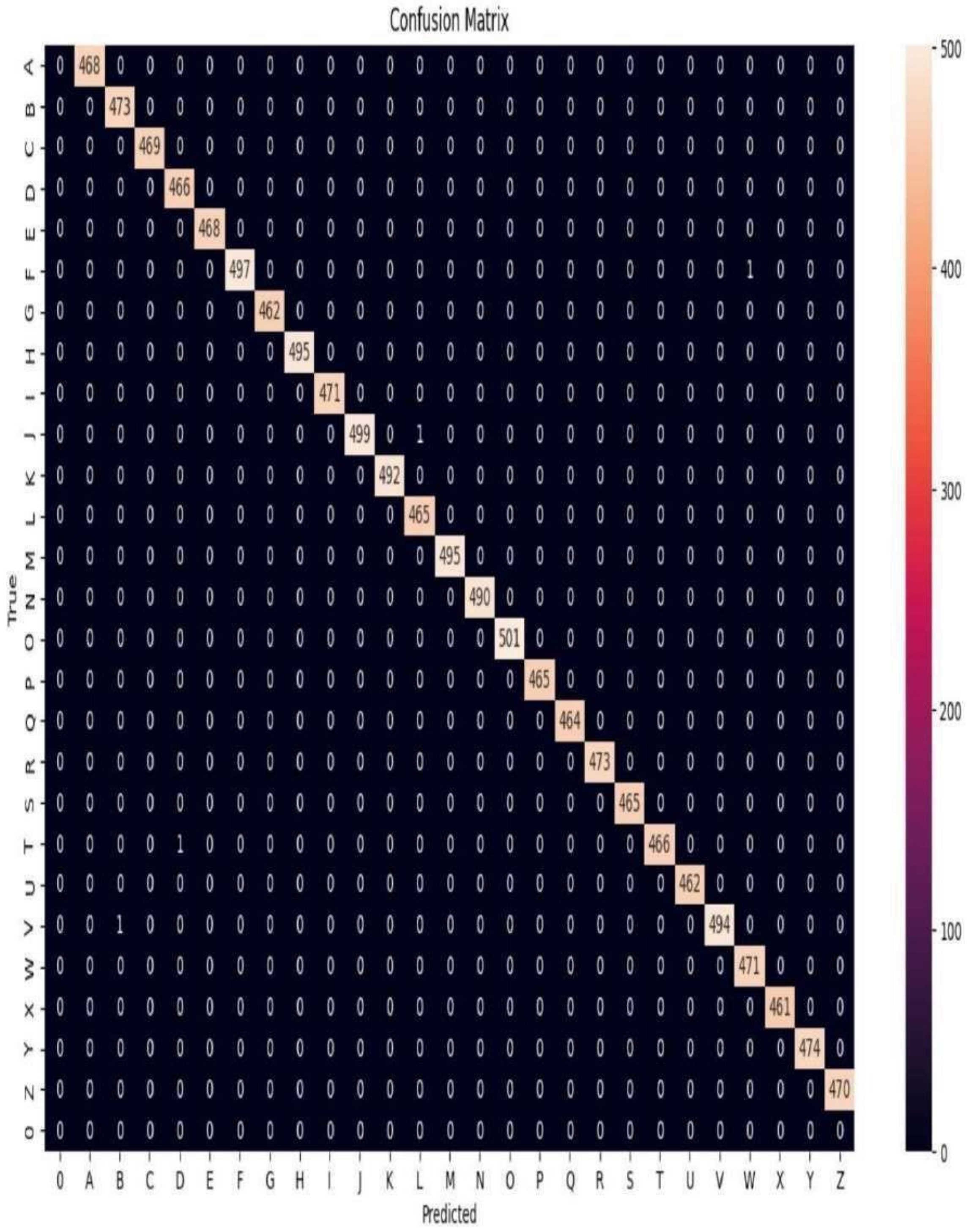


Fig 6.6. Confusion Matrix

REFERENCES

[1] A. S. Nandhini, D. Shiva Roopan, S. Shiyaam, and S. Yogesh, “Retraction: Sign Language Recognition Using Convolutional Neural Network,” Journal of Physics: Conference Series, vol. 1916, no. 1. IOP Publishing Ltd, May 27, 2021. doi: 10.1088/1742-6596/1916/1/012091.

[2] Y. Dhamecha, R. Pawar, A. Waghmare, and S. Ghosh, “Sign Language Conversion using Hand Gesture Recognition,” 2023 2nd International Conference for Innovation in Technology, INOCON 2023, 2023, doi: 10.1109/INOCON57975.2023.10101099.

[3] S. Shivdikar, J. Thakur, and A. Agarwal, “Hand Gesture Recognition and Translation Application,” International Research Journal of Engineering and Technology, 2022, [Online]. Available: www.irjet.net.

[4] A. Rathi, S. Pasari, and S. Sheoran, “Live Sign Language Recognition: Using Convolution Neural Networks,” 8th International Conference on Advanced Computing and Communication Systems, ICACCS 2022, pp. 502– 505, 2022, doi: 10.1109/ICACCS54159.2022.9785357.

[5] H. Limaye, S. Shinde, A. Bapat, and N. Samant, “Sign Language Recognition using Convolutional Neural Network with Customization,” SSRN Electronic Journal, Jul. 2022, doi: 10.2139/SSRN.4169172.

[6] M. Naveenkumar, S. Srithar, G. R. Kalyan, E. Vetrimani, and S. Alagumuthukrishnan, “Hand Sign Recognition using Deep Convolutional Neural Network,” 4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings, pp. 1159–1164, 2022, doi: 10.1109/ICIRCA54612.2022.9985691.

[7] K. K. Dutta and S. A. S. Bellary, “Machine Learning Techniques for Indian Sign Language Recognition,” International Conference on Current Trends in Computer, Electrical, Electronics and Communication, CTCEEC 2017, pp. 333–336, Sep. 2018, doi: 10.1109/CTCEEC.2017.8454988.

[8] P. Rai, A. Alva, G. K. Mahale, J. S. Shetty, and M. A. N, “International Journal of Computer Science and Mobile Computing GESTURE RECOGNITION SYSTEM,” 2018. [Online]. Available: www.ijcsmc.com

[9] J. L. Crowley, ACM Digital Library., and ACM Special Interest Group on Computer-Human Interaction., Proceedings of the 2009 international conference on Multimodal interfaces. ACM, 2009.

[10] E. A. Kalsh and N. S. Garewal, “Sign Language Recognition System.” [Online]. Available: www.ijceronline.com.

[11] Brill R. 1986. The Conference of Educational Administrators Serving the Deaf: A History. Washington, DC: Gallaudet University Press.

[12] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278- 2324, Nov. 1998.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[14] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.

[15] Kang, Byeongkeun, Subarna Tripathi, and Truong Q. Nguyen. ”Real- time sign language fingerspelling recognition using convolutional neural networks from depth map.” arXiv preprint arXiv: 1509.03001 (2015).

[16] Suganya, R., and T. Meera devi. ”Design of a communication aid for physically challenged.” In Electronics and Communication Systems (ICECS), 2015 2nd International Conference on, pp. 818-822. IEEE, 2015.

[17] SruthiUpendran, Thamizharasi. A,” American Sign Language Interpreter System for Deaf and Dumb Individuals”, 2014 International Conference on Control, Instrumentation, Communication.

[18] David H. Wolpert, Stacked generalization, Neural Networks, Volume 5, Issue 2, 1992. 46

[19] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, Volume 12, Issue 10,1999.

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine Update

journal homepage: www.sciencedirect.com/journal/computer-methods-and-programs-in-biomedicine-update

DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals

Ahmed KASAPBAS¸ I [a], Ahmed Eltayeb AHMED ELBUSHRA [a], Omar AL-HARDANEE [a], Arif YILMAZ [b,*]

[a] Department of Electrical and Electronics Engineering, Graduate School of Natural and Applied Science, Ankara Yildirim Beyazit University, Ankara, Turkey
[b] Maastricht University, Institute of Data Science, Maastricht, Netherlands

A R T I C L E I N F O

Keywords:
Deep learning
Convolutional neural network (CNN)
Sign language recognition (SLR)
OpenCV
New Dataset

A B S T R A C T

Background: Sign language is an essential means of communication for hearing-impaired individuals.
Objective: We aimed to develop an American sign language recognition dataset and use it in the deep learning model which depends on neural networks to interpret gestures of sign language and hand poses to natural language.
Methods: In this study, we developed a dataset and a Convolutional Neural Network-based sign language interface system to interpret gestures of sign language and hand poses to natural language. The neural network developed in this study is a Convolutional Neural Network (CNN) which enhances the predictability of theAmerican Sign Language alphabet (ASLA). This research establishes a new dataset of the American Sign Lan- guage alphabet which takes into consideration various conditions such as lighting and distance.
Results: The dataset created in this study is a new addition in the field of sign language recognition (SLR). This dataset may be used to develop SLR systems. Furthermore, our research compares the results of our dataset with two different datasets from other studies. The other datasets have invariant scene conditions, but our suggested CNN model demonstrated high accuracy for all the tested datasets. Despite the different conditions and volume of the new dataset, it achieved 99.38% accuracy with excellent prediction and small loss (0.0250).
Conclusions: The proposed system may be considered a promising solution in medical applications that use deep learning with superior accuracy. Moreover, our dataset was created under variable conditions which increasesthe number of contributions, comparisons, results and conclusions in the field of SLR and may enhance such systems.

Introduction

More than 5% of the world's population is affected by hearing impairment. To overcome the challenges faced by these individuals, various sign languages have been developed as an easy and efficient means of communication. Sign language depends on signs and gestures which give meaning to something during communication [1]. Researchers are actively investigating methods to develop sign language recognition systems, but they face many challenges during the implementation of such systems which include recognition of hand poses and gestures. Furthermore, some signs have similar appearances which add to the complexity in creating recognition systems [3,4]. This paper focuses on the sign language alphabet recognition system because the letters are the core of any language [2]. Moreover, the system presented here can be considered as a starting point for developing more complex systems.

There are two types of sign language recognition methods namely sensor-based and image-based. The first method is dependent on localized sensors or wearing specific gloves. The main benefit of this method is its ability to provide accurate information about signs or gestures such as the movement, rotation, orientation as well as positioning of the hands [6–10]. The second method uses different types of cameras. It is based on image processing which does not require equipment such as sensors. This approach only relies on the application of various image processing techniques and pattern recognition [5,10-14][19].

Sign language differs among countries. In addition, various sign

* Corresponding author.
E-mail addresses: 185105405@ybu.edu.tr (A. KASAPBAS¸ I), 195105116@ybu.edu.tr (A.E.A. ELBUSHRA), 185105406@ybu.edu.tr (O. AL-HARDANEE), a.yilmaz@maastrichtuniversity.nl (A. YILMAZ).

https://doi.org/10.1016/j.cmpbup.2021.100048
Received 21 June 2021; Received in revised form 13 October 2021; Accepted 29 December 2021
Available online 10 January 2022

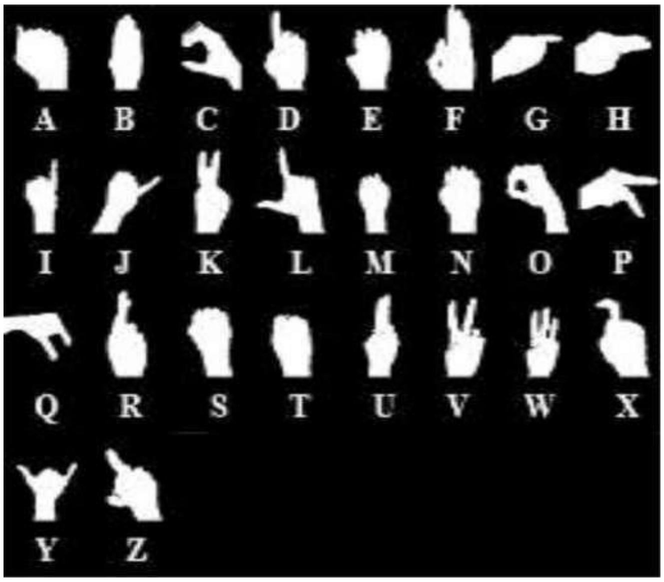


Fig. 1. Threshold hand poses of ASLA.

languages generally contain non-manual signs such as body gestures and facial expressions. They often require two hands or sequential movements for performing these signs. These issues increase the complexity in design of sign language recognition systems. To overcome this, researchers showed a specific interest in the sign language recognition systems [15].

In recent years, researchers have employed deep learning for sign language recognition systems. This involves the use of various methodologies and datasets which aim to improve the accuracy of the system. Various datasets are created because of many factors such as regional differences, type of images (RGB or Depth) and so on. Sign language differs from one region to another just like spoken languages and in- cludes American sign language, Indian sign language, Arabic sign lan- guage, etc. [3,4,12,13,14,16,17,18]. Moreover, the type of images used for recognition systems depends on the camera that generates RGB [3,4, 12,14,16,17,18] or depth images [4,13,18]. Furthermore, the methodologies used by different researchers that underlie the core of gesture recognition systems, vary from one system to another. Each study works to create and improve a different system to enhance its accuracy. Currently, there is no system which can deal with all conditions with high accuracy. Researchers have previously focused on CNNs with different parameters for sign language recognition systems due to their high performance in image classification [3,4,12,14,16,17,18]. Also, some studies use CNNs along with other methods to obtain more accu- rate results [16]. However, others have used methods such as SVM and PCANET [12,13,17]. Comparisons of CNNs with other methods have proved the superiority and ability of CNNs [12,17].

In this study, we have created an American Sign Language Alphabet (ASLA) dataset and developed a deep learning-based method for its recognition. The new dataset is designed to overcome a common challenge faced by researchers by improving the SLR system. Some chal-lenges in the recognition of the letters using the ASLA dataset[1] exist due to variation in lighting and distance. Moreover, our dataset would improve the use of methods in the fields of machine learning and deep learning by the application of different methods using this dataset and comparison of the results. The images in the created datasets were captured with laptop and smartphone cameras. The proposed method primarily focuses on static hand gestures. In this study, a CNN-based deep learning method is proposed to create a robust and real-time ASL recognition system, because CNN as a deep learning technique has shown outstanding performance in tasks of image classification and pattern recognition [1,3,4]. The CNN model described here is evaluated with our ASLA dataset as well as datasets from other research [23,24].

The rest of the paper is organized as follows. The next section contains the methodology of this study. The methodology section provides detailed information about our dataset and other datasets used in this study as well as the architecture of the proposed deep learning system. The results section presents the findings of our study. Then, the next section discusses the results. Finally, the conclusions and future work is presented.

Methodology

A. Dataset

In this study, a custom dataset was constructed to interpret all the hand poses of the American sign language alphabet from gestures tolabels. The dataset contains images and corresponding letters of ASLA. Fig. 1 shows the hand poses of the American sign language alphabet,which consists of 26 letters. In American sign language (ASL), each letter is depicted by a static sign made by using the hand to present it (excluding J and Z). The creation of the dataset was dependent on many factors such as illumination and the distance between the camera and hand which we adjusted to improve the performance of the convolu- tional neural network model. While in other datasets, the distance of the hand from the camera was reported to be fixed such as 0.5 m, 0.75 m or 1 m. Our dataset contains images varying 0.5 m, 0.75 m and 1 m hand distance. Furthermore, our dataset was collected at different times of the day specifically to include different illumination conditions. This feature is not available in the other ASL alphabet datasets. Therefore, our dataset overcomes this specific challenge for other researchers. Moreover, both the Z and J letters are created by a small movement that represents the letter, but in our study, we represent these letters by a static gesture at the start or end of the sign. This allowed us to include all the alphabet in the current dataset and obtain a complete set. Also, the required movement for these letters (Z and J) requires a separate neural network structure for achieving the recognition operation. However, our method is effective in recognizing the J and Z letters.To the best of our knowledge, there is currently no study that has reported such a method. In the field of deep learning, when a new dataset is created, it may be considered a new contribution to the field mainly because each dataset has its specific features to improve existing models. However, the availability of several datasets often creates more challenges that require solutions. Therefore, the creation of a custom dataset with special conditions may be considered as a new contribution in the field of sign language interpretation.

In addition to the custom-constructed dataset, this study evaluates two more datasets using the proposed convolutional neural network model. In each evaluation, the corresponding dataset is separated into training, validation, and testing sets. The dimensions of the images are 64×64. The first dataset includes 52,000 images of the American sign language alphabet [23]. Also, the second dataset consists of 62,400 images [24]. Finally, the custom ASLA constructed in this study consists of 104,000 images. All datasets depend on the principle of image thresholding to generate binary images. Also, the datasets are split according to the following ratios 70:15:15; 70% for training sets, 15% for validation sets and 15% for testing sets.

B. System architecture

Convolutional neural network

A convolutional neural network (CNN) is one of the most commonly used deep learning methods to analyze visual imagery. CNN involves less preprocessing compared to other image classification algorithms. The network learns the filters that are normally hand-engineered in other systems. The use of a CNN reduces the images into a format that is easier to process while preserving features that are essential for making accurate predictions. There are four types of operations in a CNN:

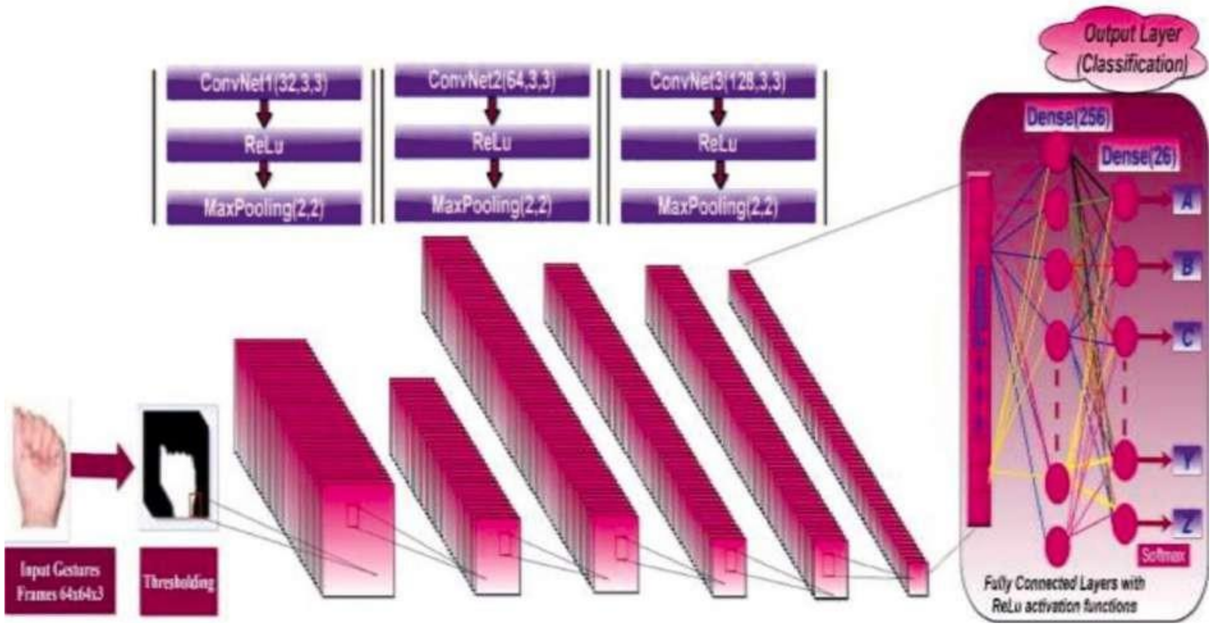


Fig. 2. The Proposed CNN model.

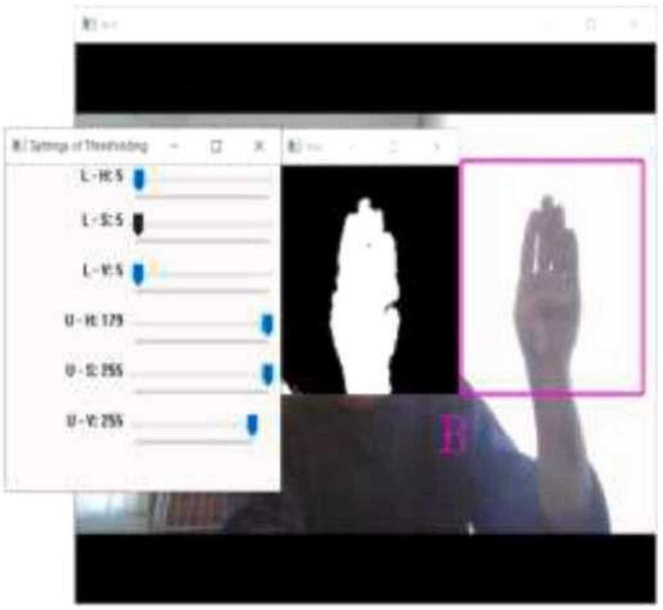


Fig. 3. The proposed prediction interfaces.

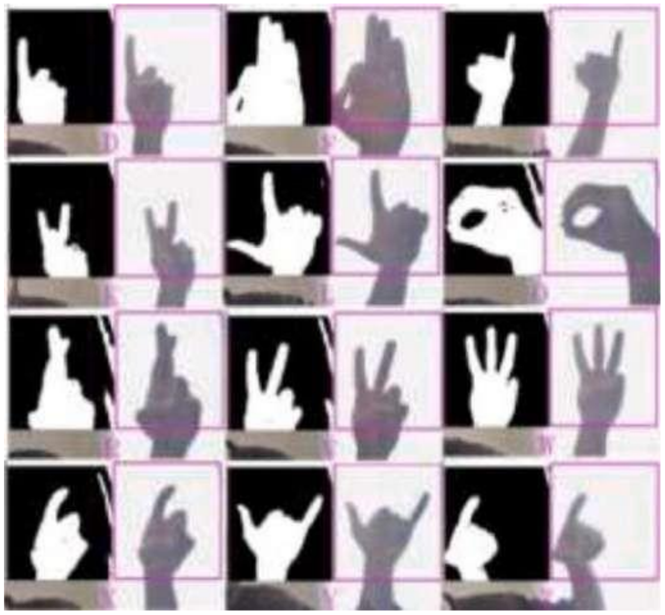


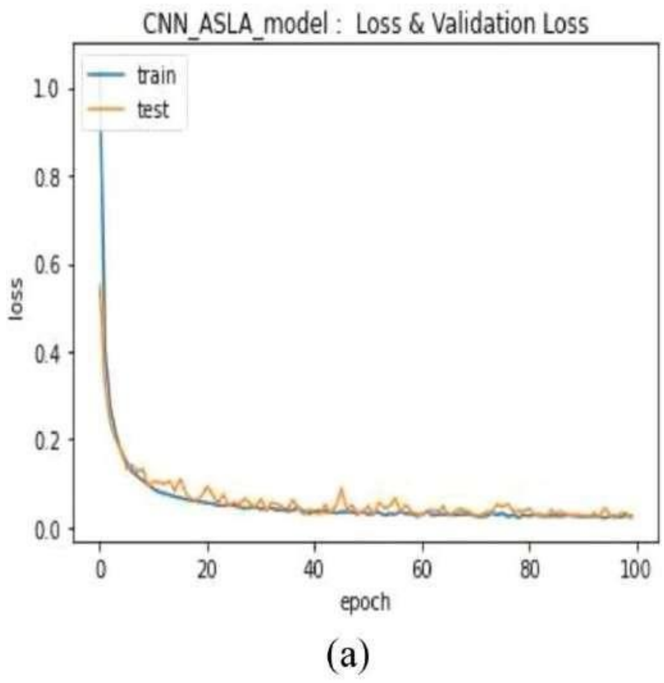
Fig. 4. Some samples of ASLA letters of this search.

convolution, pooling, flattening, and fully connected layers [20].

The convolution layer usually captures low-level features such as color, edges, and gradient orientation. The pooling layer decreases the spatial dimension of the convolved feature. This operation reduces the required computational time for dealing with the data through dimen- sionality alleviation. Furthermore, it has the advantage of maintaining dominant features that are positionally and rotationally invariant during the model training process. After the input image has been processed the higher-level features may be used for classification. Therefore, the image is flattened into a 1-D vector. In CNN, the flattened output is supplied to a fully connected layer. After training, using SoftMax classification, the model can provide probabilities of prediction of objects in the image [21].. Backpropagation is used to train the network. In this study, the system is implemented by using Keras, Tensorflow and OpenCV libraries.

The structure of the proposed CNN

The CNN model designed in our study consists of multiple layers. Fig. 2 illustrates the proposed structure of the CNN which consists of an input layer to input the images with 6×4 64×3 dimensions; this rep- resents the size of the sign language frames that are taken as input into the system. The feature extraction part comprises three convolutional layers (Conv1, Conv2, Conv3). The convolution filter dimensions in each layer are 3 ×3 . The number of filters is 32 filters for ConvNet1, 64 filters for ConvNet2 and 128 filters for CovNet3. Each convolution operation is followed by rectified linear units (ReLu). After ReLu, MaxPooling is applied with 2 ×2 dimensions. Pooling aims to prevent the loss of important information when the feature is represented. After the con- volutional stage, flattening is applied for the classification stage. The classification stage is implemented with fully connected layers followed by a ReLu activation function and one SoftMax output layer [22].



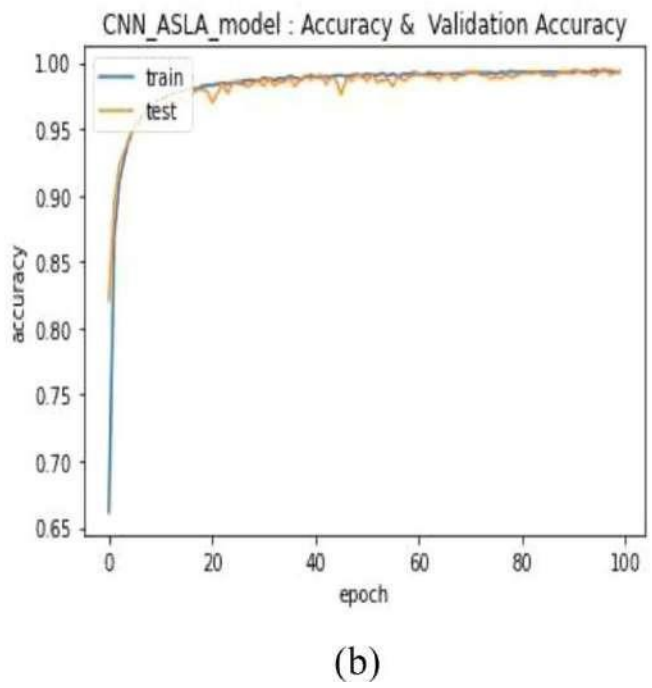
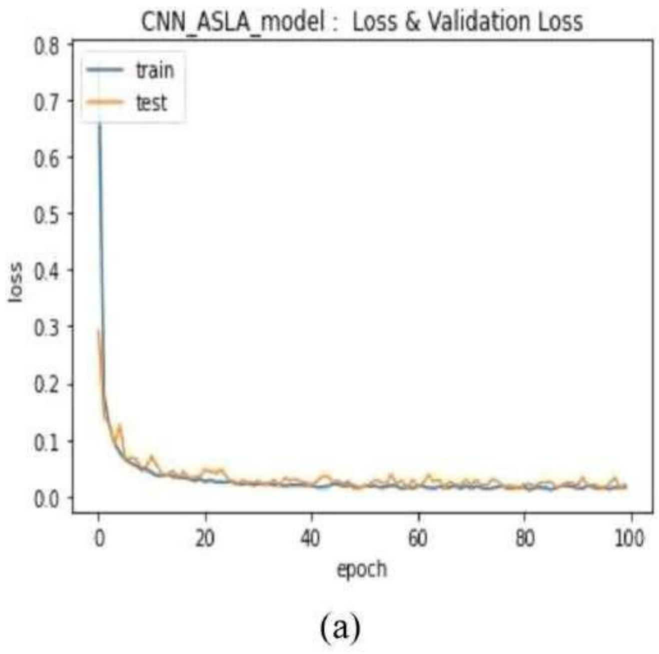


Fig. 5. (a) Training Loss and Validation Loss, (b) Training Accuracy and Validation Accuracy for the first dataset [23].

C. Application

Several researchers have tried to use deep learning for improving sign language recognition systems. G. Anantha Rao et al. (2018) pro-poses a CNN for the recognition of the gestures of the Indian sign lan- guage [3]. They used a continuous capture method from smartphone cameras in selfie mode and built a mobile application. They also generated a dataset from five different subjects performing 200 signs in5 various viewing angles under different backgrounds due to the lack of availability of datasets on mobile selfie sign language. Moreover, they trained a CNN with 3 various sample sizes. Each one consisted of numerous subjects and viewing angles. The remaining 2 samples were utilized for testing the trained CNN. As a result, the researchers obtained 92.88% accuracy in comparison to other classifier models reported on the same dataset [3].



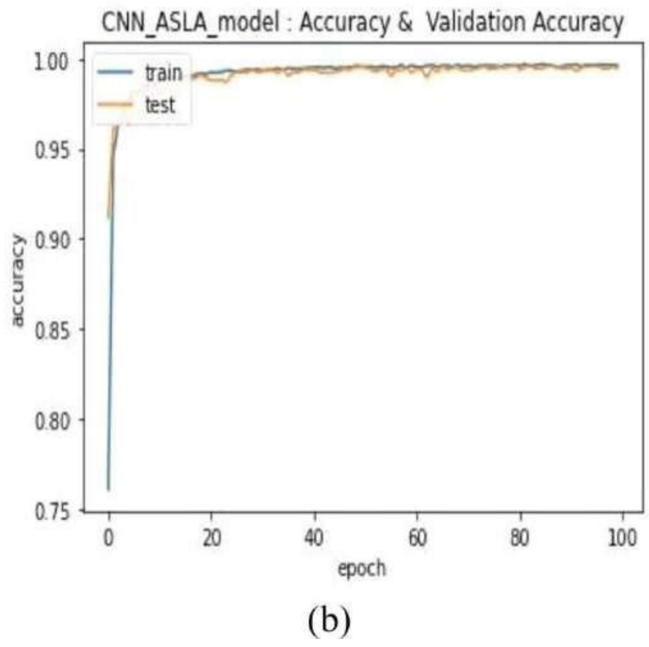
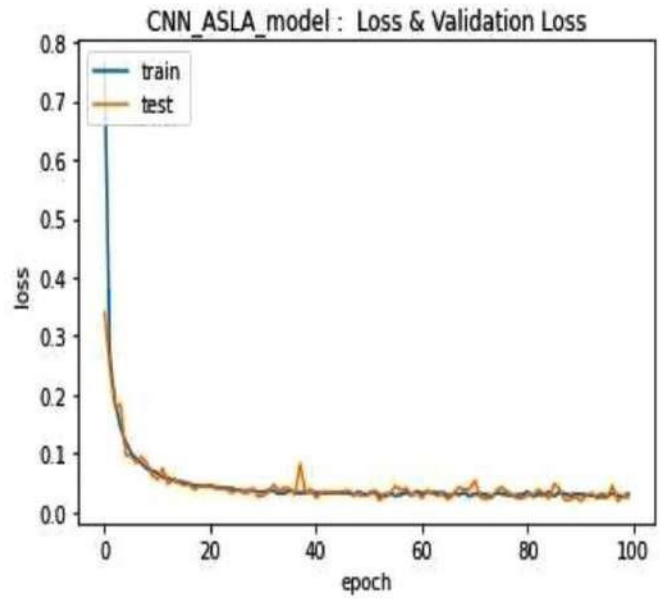
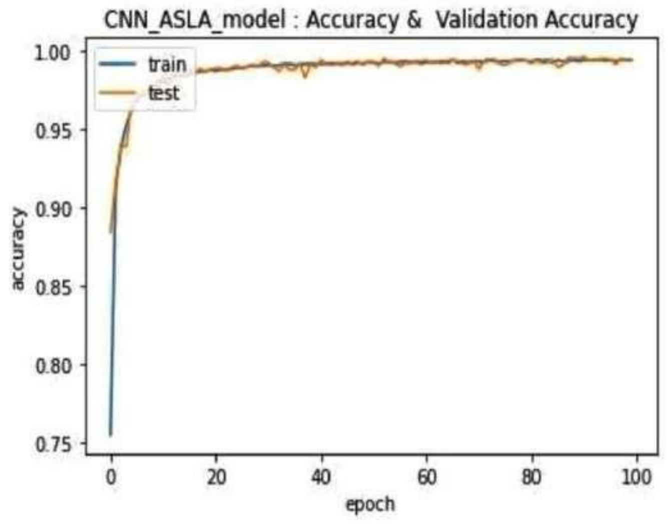


Fig. 6. (a) Training Loss and Validation Loss, (b) Training Accuracy and Validation Accuracy for the second dataset [24].

R. Daroya et al. (2018) reported a method which is CNN inspiredcalled Densely Connected Convolutional Neural Networks (DenseNet) to classify RGB (Red_Green_Blue) images of static letter hand gestures in Sign Language [4]. They also utilized a web camera in real-time to achieve their work. DenseNet has advantages including the alleviation of a vanishing gradient. The advantages of DenseNet have been used widely for classification tasks. This proposed deep network is useful for sign language classification tasks and has an accuracy of 90.3%. In addition, they use both depth images and RGB images. V. Jain et al. (2021) created a system to recognize ASL by using both Support Vector



(a)



(b)

Fig. 7. (a) Training Loss and Validation Loss, (b) Training Accuracy and Validation Accuracy, for our dataset (ASLA).

Machine (SVM) and Convolutional Neural Network (CNN) [12]. They then compared the accuracy of the two methods. They found that the single layer CNN yielded 97.344% accuracy and 98.581% accuracy was reported for a two-layer CNN. In contrast, the SVM gave 81.49% accuracy when combined with the "Poly" kernel. They further aimed to enhance the accuracy of the CNN by determining the best size of the filter. S. Alyl. et al. (2017) designed a system for alphabetic Arabic sign language recognition by using intensity and depth images, which were obtained from a SOFTKINECTM sensor [13]. They utilized a PCANet method to learn local features from intensity and depth images and a linear support vector machine classifier to recognize the extracted features. The performance of the suggested system was assessed on a dataset of real images captured from multi-users. They implemented the tests in two ways; the first one utilized intensity and depth images separately, the second one utilized a combination of intensity and depth images. The acquired results revealed that the second test produced an accuracy of 99.5%. P. Kurhekar et al. (2019) present a system that can learn signs from videos by processing the video frames under a minimum disturbed background [14]. After that, the obtained sign is converted to readable text. This system utilizes a CNN which depends on a ResNet-34 CNN classifier, Fast.ai and OpenCV libraries for webcam inputs and displaying the predicted signs. They obtained a model with an accuracy of 78.5% on the presented testing set. K. Bantupalli and Y. Xie (2019) implemented a vision-based application that provides sign language translation into text in their work [16]. They aimed to aid people with hearing disabilities to communicate with hearing individuals. Their suggested model extracts temporal and spatial features from video sequences. Besides, they utilize a CNN for recognizing spatial features and a Recurrent Neural Network (RNN) to train on temporal features. They train the CNN and RNN models independently. They used thecross-entropy cost Adaptive Moment Estimation (Adam) function to minimize loss. Also, they double the size of their test dataset to gather more information from predictions about the model. In this system, the utilized dataset is the American Sign Language dataset, and the accuracy is 91% for 150 signs at the output of the softmax layer.

H. B.D Nguyen and H. Ngoc Do (2019) reported a sign language fingerspelling alphabet identification system [17]. In this study, three techniques are used for improving the system. In the first model, they utilize the combination of Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP) features and multi-kernel multi-class Support Vector Machines (SVM) to increase the special attributes of each feature type to the maximum point. Specifically, the researchers use 24 alpha- betical symbols presented by static gestures, but they exclude two mo- tion gestures which are the J and Z letters because of the dynamic structure of these letters. HOG and LBP features of each gesture are extracted from training images. After that, they train these extracted data by multi-class SVMs. The researchers also utilize an end-to-end CNN architecture for comparison. The combination of CNN as a feature descriptor and SVM generated an acceptable result. The feature-kernel pairs: HOG and RBF, LBP and Polynomial have anaverage recognition rate higher than two methods that utilize only one feature (98.36%). The CNN and CNN-SVM models provide 97.08% and 98.30% accuracy, respectively, which provides evidence that by executing CNN as a standalone feature extract, better results could be obtained than utilizing other CNN architecture. Despite the lower ac-curacy of the CNN-SVM model than that of the HOG-LBP- SVM model, it has a better opportunity when facing overfitting. S. Ameen and S. Vadera (2016) improve a convolutional network which they achieved by applying it to the problem of fingerspelling recognition for ASL [18]. This method improved the CNN model's classification performance in fingerspelling images by using both image intensity and depth data, introducing the applicability of deep learning for interpreting sign language and development. Their results showed that the improved model had 82% precision and 80% recall. Furthermore, they evaluated the confusion matrix and identified difficulties in classifying some signs.

In this study, we also developed an application for American sign language alphabet recognition in real-time. The OpenCV library was used to capture video due to its performance and robustness. The capture resolution is 640×480. Moreover, three windows appear when the application is running in prediction mode. Fig. 3 shows three windows of the application. The first window is the image capture window which contains a predefined region with dimensions of 196×196. The user is required to perform the signs in the predefined region. The application captures images with 64 6 ×4 dimensions which is necessary for the CNN model. After capturing the image from the predefined region, the thresholded image is displayed in the preview window. Then, the application saves the threshold display capture images to supply to the CNN model for classification. Finally, the predicted character appears under the predefined region of the image capture window. The control window with track bars is used for controlling the threshold. Fig. 4 illustrates some samples which were predicted by the program for

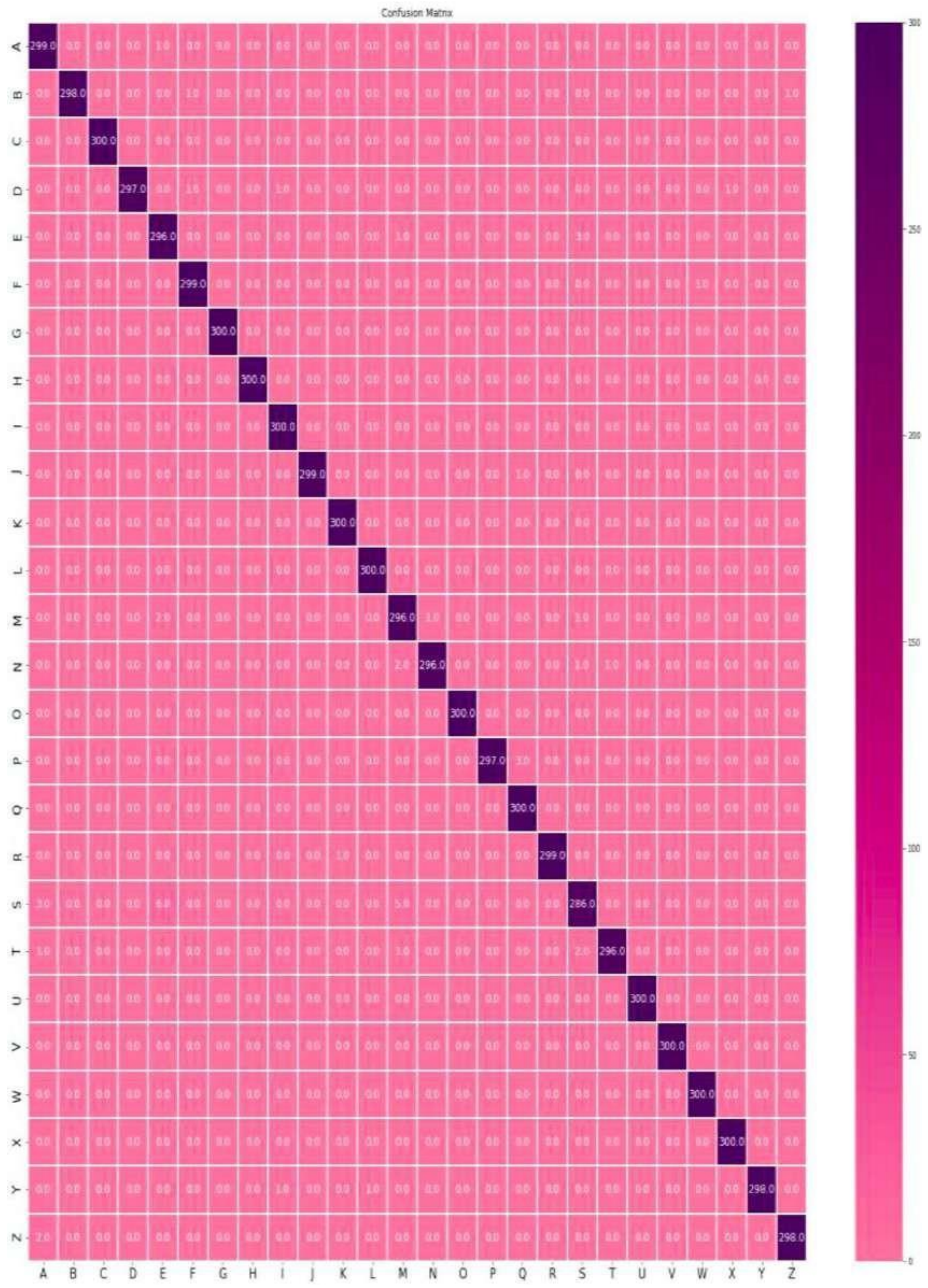


Fig. 8. Confusion matrix of test dataset for the first dataset in [23].

different American sign language Alphabet letters and the Z letter appears as a static letter at the start point of the required movement.

Results

Three separate American Sign Language Alphabet datasets were used to evaluate the proposed CNN model. These datasets are described in the methodology section. According to the results of the experiments, the proposed CNN model outperformed other neural network structures in certain metrics. The three datasets were applied to the proposed CNN for 100 epochs according to the following: Firstly, the training was executed for the first dataset [23] and the obtained accuracy was 99.41% with a 0.0204 loss. Secondly, the training was implemented to the second dataset [24], for which the obtained accuracy was 99.48% and the loss was 0.0210. Finally, we trained the suggested CNN to our dataset (ASLA). Here,

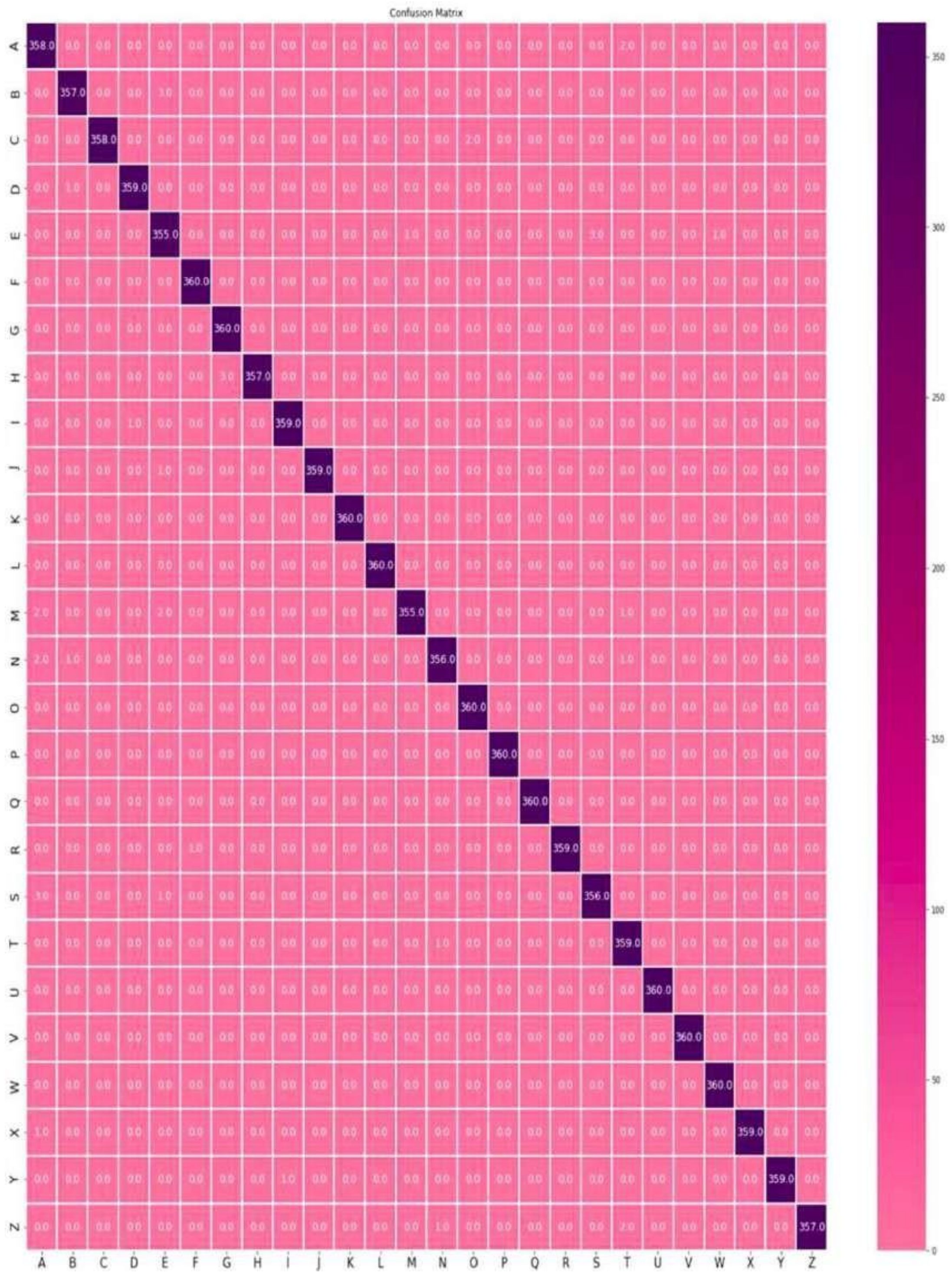


Fig. 9. Confusion matrix of test dataset for the second dataset in [24].

the obtained accuracy was 99.38% and the loss was 0.025. In this case, the dataset was larger than the two others. Figs. 5, 6 and 7 show the model losses which represent the training loss versus validation loss in part (a), and the model accuracywhich shows the training accuracy versus the validation accuracy in part (b) for the first, second and our (ASLA) dataset sequentially. Moreover, Figs. 8, 9, and 10 represent the confusion matrix of the test dataset for the first dataset [23], second dataset [24] and ALSA dataset sequentially. The confusion matrices illustrate the high performance of the suggested model. Table 1 illustrates a comparison of the accuracies and losses the CNNs applied on the three different datasets.

Although the experimental results show that the first and second datasets have an accuracy higher than our dataset, the newly generated dataset is more challenging because it relies on many different condi- tions and factors. Furthermore, our dataset was the largest one among all the other datasets and contains 104,000 images which ultimately led to the superior prediction.

Discussion

This research presents a new dataset of the American sign language alphabet. The dataset which we have created takes into consideration

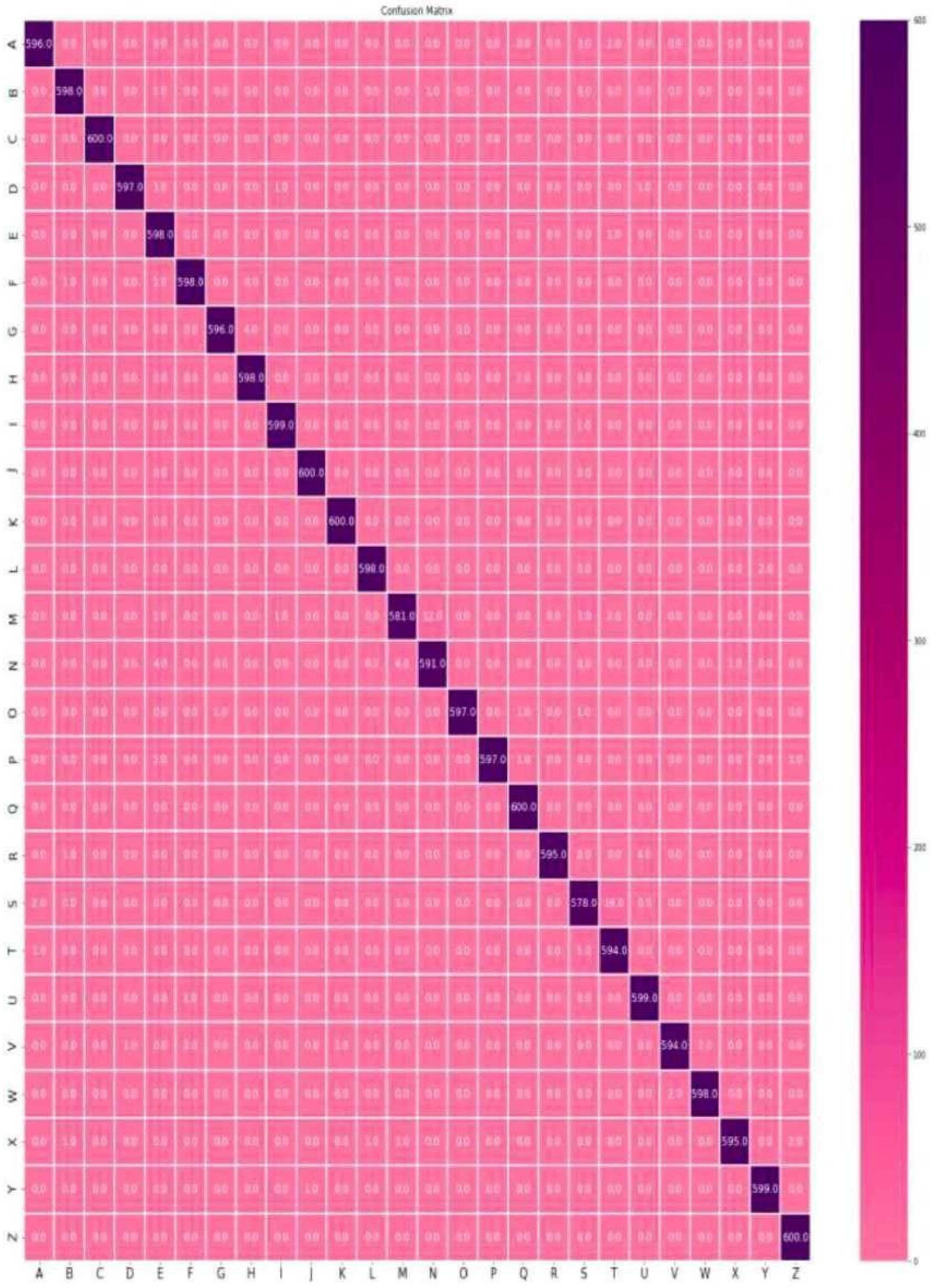


Fig. 10. Confusion matrix of test dataset for our dataset (ASLA).

Table 1

Comparison of the accuracies and for the various dataset.

The Proposed CNN Trained datasets	Accuracy	Loss
The first dataset [23]	99.41%	0.0204
The second dataset [24]	99.48%	0.0210
Dataset of this research (ASLA) [25]	99.38%	0.0250

various conditions such as lighting and distance, which makes it superior to other datasets which used non-variable conditions. This dataset is a new addition to other datasets in the field of SLR, which can assist researchers and students in developing SLR systems. Furthermore, our work presents a CNN that performs at a high accuracy under various conditions. Moreover, in our study, we have compared the accuracy of our dataset with those from other studies. Despite the different condi- tions and volume of the new dataset, it achieved 99.38% accuracy with

Table 2
Comparison of the accuracy between the suggested method and the methods in the other studies.

Method	Accuracy
CNN [3]	92.88%
DNN [4]	90.3%
CNN [12]	98.581%
SVM [12]	81.49%
PCANet [13]	99.5%
ResNet-34 CNN [14]	78.5%
CNN and RNN [16]	93%
CNN [17]	97.08%
CNN-SVM [17]	98.30%
Our CNN for three datasets	99.41% [23]
	99.48% [24]
	99.38% (ours)

better prediction and a small loss (0.0250). The comparison shown in Table 1 emphasizes the importance and contribution of our work and highlights the superior and competitive performance of our dataset. Table 2 compares the accuracy of our proposed approach with the ac- curacy of other methodologies used in other studies. Our method proves its ability to give superior accuracies for different datasets. For example, the methods used by several other researchers [3,4,12,13,14,16,17] reported accuracies that are seen in Table 2.

Conclusion and future work

Sign language is frequently used for communication by individuals who have a hearing impairment. As a result, the development of sign language recognition systems has a significant impact on these in- dividuals. For this reason, a large amount of research has been attempted to develop a device that can automatically recognize the physical gestures of sign language. There have been numerous systems designed to interpret signs from images, but many problems are stillfaced by researchers in the field. Such challenges include the variation in size, position, shape and background of the hand, lighting, and the distance of the hand from the camera. Many studies have focused on the creation of sign language recognition systems by combining feature extraction methods with classification methods to identify hand poses.In this study, we developed a CNN architecture to interpret the Amer- ican sign language alphabet. All layers have the same filtering window sizes of 3 3, which improves the speed and×accuracy of recognition,but have an increased computational time. Additionally, max-pooling was used after each convolution layer. This study utilized three con- volutional layers since increasing or decreasing the number of layersaffects the performance of the neural network.

The CNN model that we have designed has three convolutional layers since we found this number of layers provided the best accuracy in empirical trials. Three datasets have been used for evaluation. Two datasets were from other studies to measure the loss and accuracy of the designed model for each dataset. The third dataset was created in-house in this study. This new dataset may support future research in the field of machine learning and deep learning to develop sign language recogni- tion systems. The accuracy of the two previously acquired datasets is 99.41% with 0.0204 loss for the first dataset and 99.48% with 0.0210loss for the second dataset, while accuracy of 99.38% with 0.025 losswas obtained with our dataset. In real world tests, these accuracies seem competitive, but the most accurate prediction was obtained by our dataset because our dataset is larger than others. Therefore, the CNN architecture that we have presented yields higher performance than previous SLR models.

This study can be improved by adding more images for more letters and words into the dataset. Also, more images can be added to improve accuracy and reduce loss. By the addition of new words and terms, the proposed system may be improved to predict a complete word. Addi- tionally, these predicted words can be turned into speech by utilizing a text-to-speech engine. Other future work may be carried out aiming to convert the ASL to commands which robots or machines can understand and execute. In this situation, any person may stand in front of the robot and give commands to it by ASL.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] M. Mohandes, J. Liu, M. Deriche, A survey of image-based Arabic sign language recognition, in: 2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14), 2014, pp. 1–4, https://doi.org/10.1109/SSD.2014.6808906.

[2] C. Padden, D.C. Gunsauls, How the alphabet came to be used in a sign language, JSTOR 4 (2003) 10–33, https://doi.org/10.1353/sls.2003.0026.

[3] G.A. Rao, K. Syamala, P.V.V. Kishore, A.S.C.S. Sastry, Deep convolutional neural networks for sign language recognition, in *2018 Conference on Signal Processing And Commun. Eng. Syst. SPACES* (2018), 2018, vol. 2018-Janua, pp. 194–197, 10.1109/SPACES.2018.8316344.

[4] R. Daroya, D. Peralta, P. Naval, Alphabet sign language image classification using deep learning, in IEEE Region 10 Annual Int. Conference, Proceedings/TENCON (2019) vol. 2018-Octob, no. October, pp. 646–650, 10.1109/TENCON.2018.8650241.

[5] A. Thongtawee, O. Pinsanoh, Y. Kitjaidure, A novel feature extraction for American sign language recognition using webcam, in The 2018 Biomed. Eng. Int. Conference (2018) 5–9, https://doi.org/10.1109/BMEiCON.2018.8609933.

[6] J. Han, L. Shao, S. Member, D. Xu, J. Shotton, Enhanced computer vision with microsoft kinect sensor : a review, IEEE Trans. Cybern. 43 (5) (2013) 1318–1334, https://doi.org/10.1109/TCYB.2013.2265378.

[7] M. Mohandes, S. Aliyu, M. Deriche, Arabic sign language recognition using the leap motion controller, in IEEE 23rd Int. Symposium on Ind. Electronics (ISIE) (2014) 960–965, https://doi.org/10.1109/ISIE.2014.6864742.

[8] M.S. Kushwah, M. Sharma, K. Jain, Sign language interpretation using pseudo glove, in Proceeding of Int. Conference on Intelligent Commun. Control and Devices, Adva. Intelligent Syst. Comput. (2017) 9–19, https://doi.org/10.1007/978-981-10-1708-7.

[9] N. Pugeault, R. Bowden, Spelling it out : real – time ASL fingerspelling recognition university of surrey, in 3rd IAPR Asian Conference on Pattern Recognition (2011) 1114–1119, https://doi.org/10.1109/ICCVW.2011.6130290.

[10] V.N.T. Truong, A translator for American sign language to text and speech, in 2016 IEEE 5th Global Conference on Consumer Electronics (2016) 8–9, https://doi.org/10.1109/GCCE.2016.7800427.

[11] B. Kang, Real-time sign language fingerspelling recognition using convolutional neural networks from depth map, in 3rd IAPR Asian Conference on Pattern Recognition (2015) 136–140, https://doi.org/10.1109/ACPR.2015.7486481.

[12] V. Jain, A. Jain, A. Chauhan, S.S. Kotla, A. Gautam, American sign language recognition using support vector machine and convolutional neural network, Int. J. Inf. Technol. 13 (2021) 1193–1200, https://doi.org/10.1007/s41870-021-00617-x.

[13] S. Aly, B. Osman, W. Aly, M. Saber, Arabic sign language fingerspelling recognition from depth and intensity images, in 12th Int. Comp. Eng. Conference, ICENCO 2016: Boundless Smart Societies (2017) 99–104, https://doi.org/10.1109/ICENCO.2016.7856452.

[14] P. Kurhekar, J. Phadtare, S. Sinha, K.P. Shirsat, Real time sign language estimation system, in Proceedings of the Int. Conference on Trends in Electronics and Inf. ICOEI 2019 (2019) 654–658, https://doi.org/10.1109/ICOEI.2019.8862701, no. Icoei.

[15] M. Ahmed, M. Idrees, Z. Abideen, R. Mumtaz, S. Khalique, Deaf talk using 3D animated sign language, in 2016 SAI Comput. Conference (SAI) (2016) 330–335, https://doi.org/10.1109/SAI.2016.7556002.

[16] K. Bantupalli, Y. Xie, American Sign Language Recognition using Deep Learning and Computer Vision, in Proceedings - 2018 IEEE Int. Conference on Big Data, Big Data (2018) 4896–4899, https://doi.org/10.1109/BigData.2018.8622141, 2019.

[17] H.B.D. Nguyen, H.N. Do, Deep learning for American sign language fingerspelling recognition system, 2019 26th Int. Conf. Telecommun. ICT 2019 (2019) 314–318, https://doi.org/10.1109/ICT.2019.8798856.

[18] S. Ameen, S. Vadera, A convolutional neural network to classify American sign language fingerspelling from depth and colour images, Expert Syst. 34 (3) (2017), https://doi.org/10.1111/exsy.12197.

[19] K. Otiniano-Rodrıguez, E. Cayllahua-Cahuina, A. Araujo de A, G. Camara-Chavez, Finger spelling recognition using kernel descriptors and depth images, in IEEE 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI) (2015) 72–79, https://doi.org/10.1109/SIBGRAPI.2015.50.

[20] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, N U R E 521 (May 2015) 436–444, https://doi.org/10.1038/nature14539.

[21] K. He, J. Sun, Deep residual learning for image recognition, in IEEE Conference on Comp. Vision and Pattern Recognition (2016) 770–778, https://doi.org/10.1109/CVPR.2016.90.

[22] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in IEEE computer Vision and Pattern Recognition (2017) 2261–2269, https://doi.org/10.1109/CVPR.2017.243.

[23] R. Poudel, (2018, Jul 2), GitHub, "Simple-sign-language-detector". Available: https://github.com/rrupeshh/Simple-Sign-Language-Detector, (accessed 9 Feb 2021).

[24] D. Saha, (2018, May 9). GitHub, Sign-Language (Version1) Available: https://github.com/evilport2/sign-language, (accessed 9 Feb 2021).

[25] A. KASAPBAS¸I, O. AL-HARDANEE, A.E. AHMED ELBUSHRA, A. YILMAZ, (2020, January 10), GitHub, "Recognition American sign language alphabets by CNNs". Available at: https://github.com/Ahmed-KASAPBASI/Success_Team_ASL.

HAND GESTURE RECOGNITION AND TRANSLATION FOR INTERNATIONAL SIGN LANGUAGE COMMUNICATION

Sasirekha R[1], Abdul kathar J[2], Arockiyadass A[3], Naveen Raj D[4]

[1]*Assistant Professor, Department of Artificial Intelligence & Data Science, Anand Institute of Higher Technology, Affiliated to Anna University, Chennai, Tamil Nadu – 603103*

[2,3,4]*UG Student, Department of Artificial Intelligence & Data Science, Anand Institute of Higher Technology, Affiliated to Anna University Chennai, Tamil Nadu– 603103*

ABSTRACT

Hand gesture communication is one of the oldest and most natural forms of expression, particularly for individuals who are deaf or hard of hearing .Recognizing this, we have developed a real-time method using neural networks for recognizing hand gestures based on International Sign Language (ISL). Automatic human gesture recognition from camera images is an intriguing area in the development of computer vision technology. We proposes a Convolutional Neural Network (CNN) methodology to identify hand gestures associated with human actions from live camera. The primary objective is to recognize and interpret hand gestures made by individuals whose sign language as their primary mode of communication. This innovative approach not only recognizes ISL but also translates it into audible voice, thereby bridging communication gaps between those who rely on hand gestures and those who do not. It brings us one step closer to creating an inclusive environment where everyone can communicate effortlessly regardless of their auditory or verbal abilities.

ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR,OMR,KAZHIPATTUR,CHENNAI-603103

Department of Artificial Intelligence and Data Science

&

Department of Information Technology

AICTE IDEALab

GDSC

Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE

NCICDS'24

07th March 2024

CERTIFICATE

of Achievement

This is to certify that Prof/Dr./Mr./Ms Sasirekha . R

of AIHT has participated/ presented a

paper entitled Gesture voice Bridge silence Symphony Application Using CNN

in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)** organized by Department of Artificial Intelligence and Data Science & Department of Information Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu, held on 07th March 2024.

[illegible]	[illegible]	[illegible]	[illegible]
Mr.P.Sachuthanandam	**Mr.N.Manikandan**	**Dr.K.Karnavel**	**Dr.P.Suresh Mohan Kumar**
Organizing Secretary	**Organizing Secretary**	**Convener**	**PRINCIPAL**
NCICDS'24 - AI & DS	**NCICDS'24 - IT**	**HOD - AI & DS / IT**	

POORVIKA

ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR, OMR, KAZHIPATTUR, CHENNAI-603103

Department of Artificial Intelligence and Data Science
&
Department of Information Technology

AICTE IDEALab

GDSC

Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE

NCICDS'24

07th March 2024

CERTIFICATE
of Achievement

This is to certify that Prof/Dr./Mr./Ms ABDUL KATHAR.J of AIHT has participated/ presented a paper entitled Gesture voice Bridge silence Symphony Application Using CNN in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)** organized by Department of Artificial Intelligence and Data Science & Department of Information Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu, held on 07th March 2024.

[illegible]	[illegible]	[illegible]	[illegible]
Mr.P.Sachuthanandam	Mr.N.Manikandan	Dr.K.Karnavel	Dr.P.Suresh Mohan Kumar
Organizing Secretary	Organizing Secretary	Convener	PRINCIPAL
NCICDS'24 - AI & DS	NCICDS'24 - IT	HOD - AI & DS / IT	

POORVIKA

ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR, OMR, KAZHIPATTUR, CHENNAI-603103

Department of Artificial Intelligence and Data Science

&

Department of Information Technology

AICTE IDEALab

GDSC

Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE

NCICDS'24

07th March 2024

CERTIFICATE

of Achievement

This is to certify that Prof/Dr./Mr./Ms AROCKIYA DASS.A of AIHT has participated/ presented a paper entitled Gesture voice Bridge silence Symphony Application Using CNN in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)** organized by Department of Artificial Intelligence and Data Science & Department of Information Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu, held on 07th March 2024.

[illegible]	[illegible]	[illegible]	[illegible]
Mr.P.Sachuthanandam	Mr.N.Manikandan	Dr.K.Karnavel	Dr.P.Suresh Mohan Kumar
Organizing Secretary	Organizing Secretary	Convener	PRINCIPAL
NCICDS'24 - AI & DS	NCICDS'24 - IT	HOD - AI & DS / IT	

POORVIKA

ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR, OMR, KAZHIPATTUR, CHENNAI-603103

Department of Artificial Intelligence and Data Science
&
Department of Information Technology

AICTE IDEALab

GDSC

Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE

NCICDS'24

07th March 2024

CERTIFICATE
of Achievement

This is to certify that Prof/Dr./Mr./Ms NAVEEN RAJ.D
of AIHT has participated/ presented a paper entitled Gesture voice Bridge silence Symphony Application Using CNN in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)** organized by Department of Artificial Intelligence and Data Science & Department of Information Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu, held on 07th March 2024.

[illegible]	[illegible]	[illegible]	[illegible]
Mr.P.Sachuthanandam	**Mr.N.Manikandan**	**Dr.K.Karnavel**	**Dr.P.Suresh Mohan Kumar**
Organizing Secretary	**Organizing Secretary**	**Convener**	**PRINCIPAL**
NCICDS'24 - AI & DS	**NCICDS'24 - IT**	**HOD - AI & DS / IT**	

POORVIKA