

# Image Translation of Bangla and English Sign Language to Written Language using Convolutional Neural Network

by

Muttaki Islam Bismoy

18101601

Fahim Shahrear

18101451

Anirban Mitra

18101423

D M Bikash

18101609

Ferdousi Afrin

18101039

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering  
Brac University  
May 2022

© 2022. Brac University  
All rights reserved.

# Declaration

It is hereby declared that,

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain any materials previously published or written by another third party, except where it is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted or submitted for any other degree or diploma at any other university or institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

*Muttaki Islam Bismoy*

---

Muttaki Islam Bismoy  
18101601

*Fahim*

---

Fahim Shahrear  
18101451

*Anirban Mitra*

---

Anirban Mitra  
18101423

*D M Bikash*

---

D M Bikash  
18101609

*Ferdousi*

---

Ferdousi Afrin  
18101039

# Approval

The thesis titled “Image Translation of Bangla and English Sign Language to Written Language using Convolutional Neural Network” submitted by:

1. Muttaki Islam Bismoy(18101601)
2. Fahim Shahrear(18101451)
3. Anirban Mitra(18101423)
4. D M Bikash(18101609)
5. Ferdousi Afrin(18101039)

As of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering in May, 2022.

## Examining Committee:

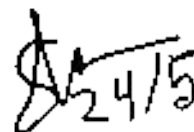
Supervisor:  
(Member)

**Hossain Arif**

---

Hossain Arif  
Assistant Professor  
Department of Computer Science and Engineering  
Brac University

Co-Supervisor:  
(Member)



---

Shaily Roy  
Lecturer  
Department of Computer Science and Engineering  
Brac University

Thesis Coordinator:  
(Member)

---

Md. Golam Rabiul Alam, PhD  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

---

Sadia Hamid Kazi  
Chairperson and Associate Professor  
Department of Computer Science and Engineering  
Brac University

## Abstract

One particular thing that differentiates humans from other species is their abilities to interact. To communicate with others, humans invented languages as units. There are 6500 Languages in this world for people of different places to communicate with each other. Among them, English has been established as a global language. As Bangladeshis, Bengali is our mother tongue and primary language to express our thoughts and feelings. However, there are a ton of physically disabled human beings who are deprived of expressing their emotions through verbal language. Therefore, Sign Language has been discovered. Expressing feelings with the help of signs is a type of Nonverbal Communication which is mainly done by moving body parts: hands in particular. Just like English, our mother Language Bengali has its own sign language consisting of 36 symbols of alphabets with its own grammar and lexicons. To resolve two way communication and a better understanding in communicating through Sign Language, in this thesis, the advantages of Real world pictures of Bangladeshi Sign Languages will be used to run an algorithm which will convert Sign Language to Written Language using Sequential Convolutional Neural Network Method. The system will be able to detect both ASL and BdSL regarding any background with the accuracy of 95.23% and 98.45% respectively.

**Keywords:** Sequential Convolutional Neural Network, Sign Language, Bangladeshi Sign Languages.

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iv</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Nomenclature</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Problem . . . . .	2
1.2 Research Objectives . . . . .	4
<b>2 Background Study</b>	<b>5</b>
2.1 Neural Networks . . . . .	7
<b>3 Literature Review</b>	<b>9</b>
<b>4 Proposed System</b>	<b>18</b>
4.1 Import Library and Mount Drive . . . . .	20
4.2 Preprocessing . . . . .	20
4.3 Generate Model . . . . .	23
4.4 Image Generator . . . . .	25
4.5 Result . . . . .	25
<b>5 Platforms, Language and Libraries</b>	<b>26</b>
5.1 Programming language . . . . .	26
5.2 Platforms . . . . .	26
5.3 Libraries and Modules . . . . .	27
<b>6 Dataset</b>	<b>29</b>
<b>7 Methodology</b>	<b>37</b>
<b>8 Result Evaluation</b>	<b>40</b>
8.1 Result . . . . .	40

8.2	Comparative Analysis . . . . .	41
8.3	Alphabetwise Accuracy . . . . .	42
8.4	Graph Analysis . . . . .	43
8.4.1	Epoch vs Accuracy graph . . . . .	43
8.4.2	Epoch vs Cross Entropy Loss . . . . .	44
8.4.3	Scatter Graph for Accuracy and Loss . . . . .	46
8.4.4	Line Model Graph (Accuracy vs Loss) . . . . .	46
8.4.5	Line Model for Validation (Accuracy vs Loss) . . . . .	47
8.5	Confusion Matrix . . . . .	49
<b>9</b>	<b>Epilogue</b>	<b>51</b>
	<b>Bibliography</b>	<b>53</b>
	<b>Appendix</b>	<b>i</b>

# List of Figures

4.1	The proposed hand gesture recognition architecture . . . . .	19
4.2	ASL established English Alphabets. . . . .	21
4.3	BdSL established Bangla Alphabets. . . . .	21
4.4	Train set for BdSL. . . . .	22
4.5	Train set for ASL. . . . .	22
4.6	Array of RGB Matrix. . . . .	23
4.7	A Neural network system taking hand gesture and passing through many convolutional layers. . . . .	24
4.8	After pooling layer, flattened as FC layer. . . . .	24
4.9	Complete process of Image recognition using CNN. . . . .	25
6.1	ঊ, এ, ও . . . . .	29
6.2	String value used for BdSL . . . . .	30
6.3	অ . . . . .	31
6.4	আ . . . . .	31
6.5	ই . . . . .	31
6.6	ঊ . . . . .	31
6.7	এ . . . . .	31
6.8	ও . . . . .	31
6.9	ক . . . . .	31
6.10	খ . . . . .	31
6.11	গ . . . . .	31
6.12	ঘ . . . . .	31
6.13	চ . . . . .	31
6.14	ছ . . . . .	31
6.15	জ . . . . .	31
6.16	ঝ . . . . .	31
6.17	ট . . . . .	31
6.18	ঠ . . . . .	31
6.19	ড . . . . .	32
6.20	ঢ . . . . .	32
6.21	ত . . . . .	32
6.22	থ . . . . .	32
6.23	দ . . . . .	32
6.24	ধ . . . . .	32
6.25	ন . . . . .	32
6.26	প . . . . .	32
6.27	ফ . . . . .	32
6.28	ব . . . . .	32



6.29	ড	32
6.30	ম	32
6.31	র	32
6.32	ল	32
6.33	স	32
6.34	হ	32
6.35	য়	33
6.36	ড়	33
6.37	ং	33
6.38	ঃ	33
7.1	Work Methodology	37
7.2	Snippet of sample code	38
8.1	Epoch vs Accuracy Graph (BdSL)	43
8.2	Epoch vs Accuracy Graph (ASL)	44
8.3	BdSL Epoch vs Model Loss	45
8.4	ASL Epoch vs Model Loss	45
8.5	Scatter Model BdSL	46
8.6	Scatter Model ASL	46
8.7	Line Model Graph (Accuracy vs Loss) BdSL	47
8.8	Line Model Graph (Accuracy vs Loss) ASL	47
8.9	Line Model for Validation BdSL	48
8.10	Line Model for Validation ASL	48
8.11	ASL Confusion Matrix Before and After Normalization	50
8.12	BdSL Confusion Matrix Before and After Normalization	50
9.1	Gesture of Alphabet “O” in the input area & taking Input in grayscale form	ii
9.2	Some noises which should have been reduced	ii
9.3	Gesture of alphabet “C” and “O” which look quite similar in grayscale form	iii

# List of Tables

8.1	Result of the applied model . . . . .	40
8.2	Comparison between applied system and other systems . . . . .	41
8.3	Alphabetwise accuracy of BdSL . . . . .	42
8.4	Alphabetwise accuracy of ASL . . . . .	42

# List of Equations

2.1 Equation of ReLU optimizer . . . . .	6
2.2 Standard (Unit) SOFTMAX function . . . . .	7
2.3 Equation of SOFTMAX optimizer . . . . .	7
2.4 Equation of ADAM optimizer . . . . .	7
2.5 Function of $\hat{m}_t$ . . . . .	7
2.6 Function of $\hat{v}_t$ . . . . .	7
4.1 Equation of flatten layer . . . . .	18
4.2 Equation of dense layer output . . . . .	18
8.2 Equation of confusion matrix's accuracy . . . . .	49
8.3 Equation of confusion matrix's precision . . . . .	49
8.4 Equation of confusion matrix's support vector . . . . .	49
8.5 Equation of confusion matrix's F1_Score . . . . .	49

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ADAM Adaptive Moment Estimation

ANN Artificial Neural Network

API Application Programming Interface

ASL American Sign Language

BASLI Bangladeshi Association of Sign Language Interpreters

BdSL Bangladeshi Sign Language

CMU Concrete Masonry Unit

CNN Convolutional Neural Network

DLBP Database system and Logic Programming

FAST Features from Accelerated Segment Test

FPN Feature Pyramid Network

HOG Head On Generation

IDE Integrated Development Environment

KNN K-Nearest Neighbour Neural Network

LDA Linear Discriminant Analysis

ML Machine Learning

PANDAS Python Data Analysis Library

PCA Principal Component Analysis

R-CNN Region based Convolutional Neural Network

ReLU Rectified Linear Activation Function

RGB Red Green Blue

RNN Recurrent Neural Network

ROI Region Of Interest  
RPN Regional Proposal Network  
SKIMAGE Sci-kit Image  
SURF Speeded Up Robust Feature  
SVM Subject Vector Machine  
SVO Subject Verb Object  
YCbCr Green Blue Red

# Chapter 1

## Introduction

Bangladesh being one of the highest populated countries in the world, has almost 228 million people speaking Bengali as their first language all over the world and beyond 37 million people speak Bengali as their 2nd Language. [1] Bengali is the 5th Native Language around the world. Not all people have the privilege to speak or hear Bengali like normal people. In Bangladesh, there are about 13 million registered deaf people and 16 million people who cannot speak. [2] People from different countries have different verbal languages that consist of different alphabets and grammar, just like that there are about 300 different sign languages all over the world for the impaired people to impart. To have a conversation with a person who has hearing and speaking impairment, understanding their sign language is very important. If the person who wants to have a conversation with an impaired person, barely has knowledge about Sign Language, it will be difficult for a two way communication. Mute people should have their own way of expressing feelings towards others. For this, an emergence of established Sign Language in Bangladesh is created. This is why the Bangladeshi Association of Sign Language Interpreter, BASLI has been established. For an effective two way communication, both the expresser and observer need to be aware of all the symbols of the Language as well as how they make words, symbolic parts of speech, grammar which is a long term process plus needs a lot of practice. Besides, this method is not effective in communication at any random places. This is how a mute or a physically deprived person remains lagged behind of a healthy two communication with any other random people. However, BASLI actually interprets the Bangla verbal and phonetic language towards the impaired people following the recognized and proposed rules of Bangladeshi Sign language, BdSL. Among all the visual languages, Sign language consists of various movements of body and hand to describe different kinds of alphabets as well as words to communicate. In Bengali Sign Language there are 30 Consonants and 6 vowels.

Furthermore, English being the most used language all over the world, American Sign Language (ASL) is an established language for mute people all over the world who mostly speak in English. ASL is a proper and sorted language which is manifested by facial and hand gestures, movements or both. ASL signs contain an amount of phonemic components for instance - movements of the torso, tongue, face and hands obviously. It was founded [3] in 1817, with 26 English alphabets and 10 numerical values. Along with different symbols for alphabets, it contains fingerspelling, iconicity and different symbols for popular words. ASL encourages

unconfined word orders where a particular syntax is not only encoded in word order, but also can be understood by other means like head nods, eyebrow movement and body position. However Sign languages mainly carry SVO formation to make a word even a sentence. Some Sign Languages have their own written form too and it is not only used in deaf community, but also used in legal recognition, telecommunication, remote interpretation etc.

Despite a lot of regulated schools for Sign Languages, not all people might feel the necessity to learn them or feel the need to ever use Sign Language in their life. Mute and Deaf people can learn sign language by reading a few books written for some particular languages. Moreover, Some common ways to communicate with impaired people are Lipsing, Home Sign, Baby Sign Language (indication), Primate use etc. So, people need a user-friendly, complete, smart and easy medium to communicate with an impaired person. Many successful systems have been created and developed by researchers all over the world. Creating a successful translation machine for sign language is not an easy task. There is an artificial neural network named CNN which is used for image recognition and processing. CNN is one kind of a deep learning algorithm which can be used to take a picture as input, assign importance such as biases, padding, kernel size, pooling, pixel density, learnable weights, spatial arrangements, parameter sharing, ReLU optimizer, Loss function, Fully Connected Layer etc to differentiate and relate between multiple images. CNN is the fastest and most accurate learning process of Image recognition.

In this thesis, it has been proposed to recognize both Bengali and English Sign Language. The proposed system will divide the image with  $128 \times 128$  pixels and then cross check it with the BdSL dataset to understand which alphabet it is. It is planned to develop an algorithm to create a meaningful word arrangement with the alphabets it has given by using Convolutional Neural Network. The application is clever enough to recognize both in Bengali and English only from hand gestures with any background at literally no cost.

## 1.1 Research Problem

As there are a handful of people who are unable to speak or hear in Bangladesh and there is no structured language, developing a conversion media can be very much helpful because if a normal person tries, he might fail due to less knowledge in the grammar of Sign Language. In this case, an automated sign language to written language converter can be the light of two way communication.

There are 6500+ languages in the world and each language has its own sign language[4]. Researchers have developed many sign language datasets, even many conversions as well. But most of them are either in popular languages or the native language of the authors. There are some open access datasets that have been developed for further research but implementing them in machine learning is an actual challenge. English is the first or second language almost all over the world. Even Bengalis use a lot of English words to make the sentence more understandable. If only a translator is developed for Bengali Written Language from Bengali

Sign language, two way communication will still be a hassle for both of the people. So, Developing a system focusing both English and Bengali Sign Language is more preferable for a proper two way communication.

Understanding Bengali language requires combined letters as well as diacritic letters. It is very challenging to convert combined letters' meaning from hand gestures. Therefore, an automated learning algorithm needs to be developed to make the sentence understandable and consequent. There are several Neural Network that can work on an image to recognize its gist part such as ANN(Artificial Neural Network), KNN (K-Nearest Neighbor), CNN (Convolutional Neural Network), SVM (Support Vector Machine), Tensorflow. ANN receives the input of images in the form of vectors and then mathematically designated by the notations of  $x(n)$  for every  $n$  number of inputs. Then distinguish images in multiple parts such as Binary, Sigmoidal Hyperbolic, Input Layer, Output Layer and Hidden Layer. ANN handles image data by two means - one by considering each and every pixel of the image as feature value and the second one to extract the image feature then feed that to ANN as feature vector. The first method miserably fails as it is unable to handle rotation, size, projection and so on. The second method needs a very good programmer to fetch the image features as it is a very tough job to do. This is why, for image processing along with image recognition ANN should not be the first choice. On the other hand, KNN detects unknown data points from common classes among the  $k$ -closest examples and calculates the distance of two data points. It actually works with a Train set and a Test set based process and has a huge accuracy while working on large datasets. Although KNN works on huge datasets, the output rate might be slow and the accuracy only depends on the quality of datasets and as it stores all the data in training, it is computationally expensive too. So, KNN is not a good choice as well. However, SVM is a binary classifier based on supervised learning which provides better performance than any other classifiers. To process images, SVM can mainly classify data into different classes and solve the regression problem and outputs a map of the sorted data with the margins between the two as far apart as possible. It does not perform well for large datasets and the train time is very high. For recognition, SVM is also not enough. Again, Tensorflow is an open source library of Google Brain team and works for deep neural networks for image recognition and classification and for a specific dataset among OpenCV or Tensorflow, Tensorflow is the best choice. But data ingestion, Tooling with Bazel, Dynamic models, profiling, having sequence inputs, creating layers, in these terms, Tensorflow fails to contribute properly. Although Tensorflow is a built in and faster to execute algorithm which provides high level libraries like Keras, it is alone still not good enough. Finally, Convolution Neural Network is a very useful way of labeling objects from images and training for supervised learning. Recognizing signs from hand gestures from an image, CNN is good at recognizing images and creating a meaningful word and it is very much particular in the process of pixel data. CNN follows a hierarchical model which works to build a fully connected layer where all the neurons are co-connected. Here, two sets of convolutional layers are set for train and test set and weight matrices and biases, filters, padding, pooling makes the task of classification and recognition easier. In terms of Images, CNN has the capability to work on feature maps and with a remarkable dataset, it is easier to learn the image pattern through CNN. CNN being a part of ANN, extracts features from images



and classifies these features with fully connected networks. Therefore, according to the study, CNN is the best choice to work with images.

## 1.2 Research Objectives

This thesis aims to convert Sign Language captured in a static image in both Bengali and English Language to written Bengali or English Language using CNN, ML and optimizers.

The main objectives of the research are as follows:

- Create a BdSL Dataset with minimum 150 pictures for each Bengali Alphabets
- To detect Bengali Sign Language two hand gestures from Dataset
- To fetch American Sign Language from Dataset
- Create a train-test split
- Fit the model for both ASL and BdSL with the same efficient fitting model
- Train the Model
- Learn to overcome errors using supervised learning algorithms
- Cross check the accuracy with random input from the test set and analyze the accuracy of the output.

# Chapter 2

## Background Study

### **Sign Language**

Any sort of communication using hands or arms or any kind of body movements when verbal communication is not possible or desirable reflects the idea of sign language. Throughout the world, there are various types of sign languages and approximate between 138 and 300 varieties of sign language that are utilized all over the world such as French sign language, British sign language, Chinese sign language and so on.[5] Even some countries have same alphabetical language but different in sign language like British sign language and American sign language. But this sign language has become a bridge between a random people and physically disabled people who face difficulties in communicating verbally.

### **Bangla Sign Language**

Sign language that is used in Bangladesh using Bengali alphabets can be defined as Bangla sign language. It is also used by the hearing or speaking impaired people in Bangladesh. For each 36 letters in Bengali alphabets, different 36 sign have been assigned, also the grammar, syntax is different. Though more than 3 million deaf people exist in Bangladesh, everyone has not enough chances to learn about BdSL and face difficulties in every aspect of life for which they are always lagging behind.[6]

### **American Sign Language**

American sign language is the natural and common language that is used by deaf community mostly in United States and Canada. Despite being a full language, with all characteristic of spoken languages, it is developed differently from natural English language. Britain and America have similar spoken language but they have approximately 30% similarities in sign language which makes them quite different to each other.[7] ASL has different kind of syntax and grammar that makes the language easier to understand. To express the ASL alphabets mostly one hand is enough to create the gesture which is quite easier than other sign language. The goal of prioritizing ASL has always been for deaf community so that they can acquire fluency in the academic and work sector.

### **Image Processing**

Nowadays one of the fastest growing technologies is image processing. Image processing is a technique in which some operations on some images is performed to improve them or to bring out relevant details from them. In this type of processing

input is an image but as an output along with images, different kind of features also can be extracted. So, this image processing can be divided into three steps. Firstly, importing an image, then the image is examined and manipulated. Finally desired image or various characteristics related to the image is extracted as an output. This image processing is followed by two methods, one is analog image processing and another is digital image processing. On analog signals, analog image processing is used and these analog signals can be both periodic or non-periodic. Electrical impulses are used to modify the images. Photographs, medical images, television images are some of examples of analog images on which analog processing is imposed. Again, on digital images digital image processing is used. A variety of software and algorithms are adapted to conduct the changes on images. Color processing, video processing are some of the examples of digital images on which digital processing is applied. Among analog and digital image processing, digital image processing is used widely and growing vastly in industries.

### Optimization

Optimization is the process by which any functions or mathematical expression can be minimized. Various optimizers such as softmax, ADAM optimizer and so on is used on minimizing the functions or reducing the errors. Optimizers are mainly the methods or strategies for reducing losses by changing the properties of neural network such as learning rate, weights etc. In a predictive modeling project, optimization occurs when data is being prepared or in model selection. In data preparation, optimization helps in transforming the raw data into a desirable or appropriate for the learning algorithms which includes scaling the data, handling missing values, normalization and so on. The goal of optimization is to come up with the optimal design based on a set of prioritized criteria or constraints where increasing production, reliability, usage, efficiency are included. Function approximation is performed on by machine learning algorithm and function optimization helps in solving the issues. Function optimization is the issue of determining the set of inputs to a goal objective function which bring out the maximum or minimum of functions and for this reason error, cost or loss is minimized while an algorithm is being fitted. ReLU optimizer, ADAM optimizer and softmax are used for the research purposes.

Rectified linear activation function or ReLU defines the linear function that works as if the input is positive, it gives output as input directly, on the other hand if the input is negative, output becomes zero. This optimizer is easy to train the dataset and even performs better in many cases for which it is used widely for many functions like in evolving multi-layer perception and CNN. The function:

$$f(x) = x^+ = \max(0, x) \quad (2.1)$$

Another one is softmax optimizer that is basically normalized exponential function. In polynomial logistic regression it is applied to normalized the result to a probability distribution. This function works with a vector  $z$  of  $K$  real numbers that sum to 1. The inputs can be zero, positive or negative where this function converts those value between 0 to 1 that scales the values into probabilities. The standard (unit)

softmax function:

$$\sigma : R^K \rightarrow (0, 1)^K \quad (2.2)$$

which is defined when K is greater than 1 by the formula:

$$\sigma(\vec{Z}) = \frac{e^{Z_i}}{\sum_{j=1}^K e^{Z_j}} \quad (2.3)$$

Here,  $i = 1, \dots, K$

$z = (z_1, \dots, z_k) \in R^K$

$\sigma$  =softmax

$\vec{Z}$ =input vector

$e^{Z_i}$ =standard exponential function for input vector

$K$ =number of classes in the multi-class classifier

$e^{Z_j}$ =standard exponential function for output vector

ADAM optimizer or adaptive moment estimation is an algorithm that combines two gradient descent methodologies with a learning rate for each network parameter. This is used to speed up the gradient descent algorithm by considering the gradient's exponentially weighted average. Using averages accelerates the algorithm's convergence to the minima. While working on a large data this function works efficiently but it uses very less memory. It also works great in very noisy problems.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (2.4)$$

Here,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.5)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.6)$$

$\beta_1 = 0.9$

$\beta_2 = 0.999$

$\epsilon = 10^{-8}$

$\hat{m}_t$  = average of past gradients.

$\hat{v}_t$  = average of past squared gradients.

## 2.1 Neural Networks

### CNN

In this system CNN has been used because of the hidden layers. The hidden layers filter out the image in different layers and give us an outcome. CNN creates the matrix value of an image with its pixel. For ASL, the pictures in dataset have been converted into 64\*64 pixels and in case of BdSL, the pictures in dataset have been

converted into 128\*128 pixels. To filter the image out, an ideal matrix of the hand has run over and detects the hand section of the picture. This ideal matrix comes from the matrix value of datasets and the matrix values are the pixel values of every picture. For instance, from the dataset's train data the system can know about the matrix value of the picture. From that value, the pictures' hand area has been identified and learned the matrix value of that area. When the test data has been given, the matrix value of the hand can be identified and the system can detect this part as hand. This matrix identification has been done through the hidden layer of the CNN. In the beginning of the system has been declared to run on 64\*64 matrix size. The Kernel size is 4. The number of strides has been given 1 and the optimizer is ReLU. After that the same thing has happened but with stride value as 2. Then the Dropout regularization method has been applied to prevent overfitting. Then the same process are going to happen of 128\*128 matrix, 256\*256 matrix and 512\*512 matrix. The whole thing has been done to do kernel padding, Max-pooling and Flattening the datas.

### **Reason behind choosing CNN over other Neural Networks**

CNN is ideal when a large number of features need to be extracted from a large number of image dataset [8]. In this regard, ANN (Artificial Neural Network) needs point to point data. For example, in the dataset a large number of pictures have been used with different persons and different hand positions. If the system has run through ANN then the fingers length and width are needed to be given as data points. CNN extracts these structural features from the images of the dataset. Additionally, CNN can detect the focused features of an image easily. Unlike ANN, CNN does not need to use images only converted to one dimensional vectors. This decreases the trainable parameters resulting in a decrement in storage.

The parameters in Neural Network increase promptly as the number of layers increase. This takes a huge time [9]. CNN reduces the time it takes to set the parameters. Every layer of the CNN has multiple convolution filters that process and scan the complete feature matrix and perform dimensionality reduction. This makes CNN a very convenient and appropriate network for image classification and processing. This makes CNN a neural network with higher accuracy for image classification.

RNN (Recurrent Neural Network) is another Neural Network which has some drawbacks to work with any image based system. RNN mainly works with sequential data whereas this system needs to work with image data. RNN doesn't support the system of structural relationship whereas CNN supports structural relationship. RNN has less features in compatibility when it is compared with CNN. The application side of RNN is mainly text to speech conversion but for this system, there is need of hand gesture detection which can be done well under CNN. For these reasons, CNN has been used rather than any other Neural network.

# Chapter 3

## Literature Review

M.A. Rahman et al.[10] develops a recognition technique of Bangladeshi Sign language in real time and in a variety of backgrounds where performers are also self-reliant. In this thesis, firstly the data has been collected of different Bengali alphabets and Bengali numeral signs. A dataset has been created where 10 performers act. Total 10 images of each predefined hand gesture have been found from 10 different performers. That means there have been  $(10 \times 10 \times 36)$  3600 images (alphabets) and  $(10 \times 10 \times 10)$  1000 images (numerals) for training purposes. For testing the system, 7200 images of Bengali alphabets and 2000 images of numeral signs have been used whereas 3600 images of alphabets and 1000 images of numerals are tested from other new performers. The large dataset makes the system liberated and perform better in illumination, varying environment. Using Haar cascade classifiers the system detects hand posture and after skin-color segmentation images are converted to binary images, resized with  $150 \times 150$  resolutions. Because of background variation, the system has used morphological operation and Gaussian smoothing to decrease the noise. After preprocessing the data, the system has used row vector as feature vector to reduce the computational cost which helps a lot in training and identifying the image. KNN classifiers have been applied to recognize these Bengali alphabets and numerals signs with those resized images. This system has been tested under the same environment in which performers have been involved in the training process and also under different environments where other new performers participated only in the testing phase. Using the KNN, the system has achieved 92.04% accuracy for alphabets and 96.67% for numerals recognition from them who have been participating in the training stage whereas average 91.41% accuracy for alphabets and 96.39% for numerals recognition from new and non-trained performers. The computational cost is 88.09 millisecond per frame with sufficient accuracy rate. Though the system works better in disarrayed backgrounds, it has some constraints too. It fails to detect gestures if some other skin-colored object is present within the Region Of Interest (ROI) and also to compare signs which have similar binary signs. But in the near future, these limitations can be overcome and the system can be a great aid to develop the communication process among impaired and normal people.

Sanzidul et al.[11] has developed an open source dataset of BdSL using Artificial Neural Network and Machine learning. As a mute person can not hear or speak anything, they try to use different physical motions to create different signs. Authors took the help of the Bengali Sign Language Dictionary which has been published by

the Social Welfare sector of the Education Ministry in 1974. The project is the first to accomplish a successful open access Bengali Sign Language dataset. The project has been done with a sample of 1800 pictures (36\*50) of 128\*128 pixels. The system manages to capture all the 30 consonants and 6 vowels of Bengali Sign Language in White Background and cropped and resized using Python cv2 Script and then converted the picture to .jpg grayscale image. First, they have captured the images at a stable resolution and prune the noise achieved in the documentation process to improve picture grade to differentiate the actual data from the picture. Each and Every Alphabet of Bengali language has been categorized according to their similarities or symbolizing. Secondly, the captured images have been sent for cropping. The images are cropped observing both x and y axis for future processing to get maximum outcome. Only the hand region has been kept and the rest are cropped for certain purposes. Thirdly, to make the dataset more usable in Machine Learning, Deep Learning, Computer Vision based work or the Python CV2 script, images are scaled to lower pixels and RGB is set to grayscale. All images are set to one common format. A learning optimizer ADAM has been used by the researchers at a learning rate of 0.001. Adam optimizer works for 30 iterations following convolution, maxpool, dropout and dense. The model contained 9-Layer CNN for conclusion 1 and 2 where filter size is 32, Kernel Size is 5\*5 and Stride is 1\*1. The pictures have been padded in ReLU(1) activation and the output layer has been done by activating SoftMax. ReLU is followed by 2\*2 padding and 25% dropout then the dense layer is set to 256 units and dropout to 50% before final output layer and SoftMax to 36 units at the final Layer. The authors kept 15% data for testing and 85% data for training. In that paper, to perform classification and to complete the prediction task properly, a neural network architecture has been used. A Learning rate reduction method which is automatic has been used to avoid the errors and Cross Entropy function and with their implementation in the sample pictures using all these methods, the researchers have managed to get 92.65% accuracy in recognizing Bengali Alphabets from Bengali Sign Language gesture images. The Authors took the supremacy of the rapid reckoning time with an elevated learning rate as they monitored the validation accuracy. Sanzidul et al. has successfully created a complete dataset that will help everyone else's work easier who will work on BdSL.

Another system has been developed by Kohsheen Tiku [12] and the main purpose of their thesis is to convert the American Sign Language to text. An android application has been developed for this problem using a vision-based method for which general users can effectively translate sign language to text. For this researchers have used Java-based OpenCV wrapper and based on the images that are 200 X 200 pixels in RGB format the system has been developed. Histogram of Gradients has been used to extract gestures from pictures. This method is chosen so that it can identify shape as well as appearance of the picture easily through the intensity of gradients and edge detection. Here, the working methodology is that the program divides the picture in small grids and a HOG direction is assembled for the pixel within each grid. After that, The SVM, a machine learning language is used to figure out the gesture. For training the dataset three array parameters have been used and those are Detection method, kernel and dimensionality reduction type. Here researchers have used an open data set to train the SVM. Initially the system used 3000 images for each letter then reduced to 100 images for each since the SVM works on a small

data set, it is reduced to 100 definite pictures for each letter. But researchers have used a unique gesture for space for which the user can clearly express the completion of sentences. For these images there have been 27 classes among them 26 classes are for letters and other one is for space. Skin color-based segmentation and gaussian blur has been implemented for processing the data. In open-CV input is taken in BGR format and then threshold contouring has been applied for removing noise. For smoothing and defining the edges detection method, kernel and dimensionality reduction methods are applied. After processing the given input SVM is applied which identifies the image and shows the words on the screen. Then these words are converted into speech. For user interface post processing is also needed which will help the app perform more efficiently. This application also contains some features such as adding letters which indicates when other letters will be added, then a clear option can clear the whole sentences when will be needed. Like these back and speech options also help users to remove given input and convert text into speech accordingly. That is how this system can be a great help in developing the communication process.

R. Shahriar et al.[13] has developed a two-way communication between normal and deaf/dump people using a smartphone application. The application employs speech-to-sign and text-to-speech systems using CMU Sphinx framework. Sphinx is capable of defining Bengali enunciation, words, sentences and grammar. As a result, it identifies the speech that is being used as input and successfully transfigures it into phonetics text. These phonetics texts converge with themselves to form words and finally get compared with the database which is specifically created based on all the sign languages. For the speech-to-sign portion, at first it initializes the recognizer to start the process. Afterwards, it takes input from the user which is basically audio of their voices. Furthermore, the CMU sphinx speech recognizer converts voice input into text by making them in full words, breaking them down into syllables and storing them in a phonetic dictionary. In the meantime, inputs are splitted into individual words. In the end, it displays the appropriate sign linguistic output. For the text-to-speech portion, the programme checks if the internet connection is fully operational or not. After making sure that the internet connection is good enough to proceed further tasks, it takes Bengali text input. After that, it prepares the input text in such a way that it gets capable of sending to the server that is used for 'Google Translate' service as a Hypertext Transfer Protocol request. For preparing, the total length of input is calculated first. Subsequently, the sentences are diverged based on the spaces and white spaces between words. In the last stage of the preparatory phase, the summation sign is included in between the splitted words. After getting prepared for being sent and being sent as Hypertext Transfer Protocol request, it works in such a way that if the 'Play' key is clicked, the created content is broadcasted via Google Translate Server. In addition to the 'Play' key, the system also contains 'Pause' key (to stop the stream temporarily), 'Resume' key (to continue the paused stream from the specific timestamp when it is paused) and 'Replay' key (if anyone is unable to understand the speech completely or needs to remember something from the speech, he/she can play it once again using this key). For the implementation of the whole process, mainly experimental datas has been used. After the whole implementation of the process, the average accuracy is 84.71% under different circumstances such as moderately quiet places (lab), quiet



places (living room) and some areas which are full of noises and chaos.

From the research of Ahmed et al.[14], it is found that a system is developed which is Bangladeshi sign language recognition system using fingertip position using a dataset on pictures of Bengali alphabets. Also it is found that individual fingertip positions are used to understand every other symbol or alphabet. The dataset consisted of 508 pictures where there are 37 different alphabets. As a consequence, through the Artificial Neural Network, the dataset has been trained to meet the position for learning values. Ravikiran's [15] technique has been followed to detect edges and fingertip positions of every individual finger. Finding some dissimilarities, the similar kind of positions have been taken to detect the symbols. After detecting the pictures, these images have been turned into symbols by applying 2 different ANN structures. One is for BSL-FTP and another one is for Gray Scale Image. Ahmed has used two dimensional space for each finger (X-Y) and has marked each finger indexed value 1-5 with X and Y like 1X, 1Y, 2X,2Y. Researchers implemented ANN in Matlab R2015. The best test accuracy has been achieved through BSL-FTP with 100 HN. The accuracy is 98.99% with a required time of 58.22 seconds. The lowest test accuracy is 90.915 for GSI HN-100 and the required time has also been the longest. The BSL-FTP HN50 needs the lowest time among the 4 ways, but accuracy wise BSL-FTP 100 HN is better. Having been compared with other methods, BSL-FTP has the best recognition accuracy.

“Ishara-Bochon”, another open access digit dataset of BdSL which is developed by Md. S. Islam et al.[16] and the main purpose is to make the dataset available as well as most usable for further research work regarding sign language. Researchers have been preparing 100 sets of each sign digit (0,1,2,3...9) and with the processing method there are a total 1000 images in their dataset. Data Preprocessing has been divided into five different steps such as the first step is capturing the image. For capturing these images, white background has been used. After cropping those images manually, images have been freed from noise. Consequently, the images are resized by 128X128 pixels and have been converted into gray-scale. For identifying a particular image, a multilayer convolutional neural network has been used and a model has been prepared which is divided into 8 layers. ADAM optimizer has been used with 0.001 mastering rate and for error calculation categorical cross entropy and automatic Learning Rate reduction method are applied. For making the optimizer converge faster the learning rate is decreased after each epoch. Using CNN, each sign has been trained with a training dataset and has achieved 92.87% accuracy. During the validation process, the percentage of accuracy score is 92.870. In addition, while applying the model on the training data, the precisions turn into 0.9652. This dataset has broadened the resources of BdSL and will help in further research work in NLP. This will also pave the way of communication between signed and general people.

A system has been developed by Victoria A. Adewale and Dr. Adejoke O. Olamiti [17] whose main purpose is to transform American Sign Language and hand gestures to text by the help of machine learning. The image segmentation process and feature detection process have been used to reach the goal. Initially, from a live image ,the hand part is cropped. Then the cropped images are tested with the database through

knn. After matching the input with the given dataset the output has been shown. But in this case, the accuracy has been only 56% which is very poor. So some pre-processing is needed for the input. The rgb image has now converted into grayscale. This is why the extra noise has reduced. Not only that, this time the hand part can easily be detected by the algorithm. After that, the program has tried to identify the contours and outline of the hand. For that, Features from Accelerated Segment Test (FAST) algorithms and Speeded Up Robust Feature (SURF) algorithms have been applied. This time a good accuracy score has been achieved. After that the data has been ready for applying Artificial Intelligence where KNN algorithm has been used. After the input is taken by the algorithm, it has been matched with the dataset. In the dataset, the symbol that matches more with the input, has a better probability of identifying the actual letter. So taking the matching score and probability the output has shown. So the whole process has been run by some steps such as, Kinect sensors have been used for data capturing and ROI for extracting data and feature detection. Then to identify the unsupervised and supervised differences of an image researchers have used KNN. From the dataset, the feature and target has been chosen and it has been tested using KNN. Then to make higher accuracy, the FAST and SURF algorithm has been taken as preprocessing steps. For each new input it has been preprocessed and then has been checked with the dataset and now a very good result is found. It is shown that mixing the FAST algorithm and SURF algorithm with the KNN has given the best result. In this time the system has given an output with 78% accuracy which is better than the previous one.

Huda et al.[18] has done some research about Bangla Sign Language Conversion System and its posterity. Their research application consists of conveyance of regular English and Bangla Sentences to optical sign language gesture and vice versa. Starner et al.[19] developed multiple conversion models by using Hidden Markov for continuously detecting ASL in real time following two different techniques for instance, one from second person viewpoint (92% Accuracy) and the other one from a first person viewpoint (98% Accuracy). The author has decided multiple techniques after reading 22 research papers. One way communication from Bangla language to BdSL is a useful method which uses Microsoft voice Command as well as Control engine to collect the chime and transform it into distinguishable specific text. Later the content is crossed checked with the dataset and if matched, the system displays pre-deposited 3D graphical hand gestures. “Intelligent Assistant” has 82% accuracy with only 10 Bengali words and has limitations to recognize words only. One more system which can be translated Bengali Sentences through BdSL gestures using a bunch of rules and a dictionary of 1000 Bengali words. Moreover, another way of communication can be BdSL to Bangla Language which gives audio output and the system computes angles between hands to body and then matches with a prestored dataset giving an auditory output with the matched gestures. 97% accuracy is gained using 9 gestures from 20 students with 3600 images of 36 Bengali Alphabets 98.17% (Vowels) and 94.75% (Consonants). Jarman et al.[20] has also used this method and got 88.69% accuracy using 46 rotational hand gestures and tested gestures using dynamic sign as per 3 seconds 15fps videos. Deb et al.[21] has found an easy way to remove the forearm in a binary image by adding wristbands of different colors on the hands in two handed gestures communication that consist of 10 alphabets with 96% accuracy using machine learning techniques LDA and PCA. Another

proposed method is using the back propagation algorithm of ANN for learning and detection from 2000 images and achieving 99% accuracy. Huda et al. in their proposed system, divided BL to BdSL into 4 parts - taking input in texts, processing inputs based on regulated grammatical rules, searching the preprocessed texts into the database and finally the cross matched output to display in the monitor. In addition, the researchers have proposed to capture BdSL gestures in mini video clips for dataset and to recognize Bengali words using Bengali grammar and Bengali parts of speech using a translation dictionary that contains 1000 Bengali words. As +Bengali voice recognition system is not well developed yet, the researchers have used a gesture detector device such as leap motion and Kinect sensor to process the gestures on a certain time frame and preprocess the gesture, then extract feature, train and test the dataset and after recognizing text and audio output will be provided and vice versa for the opposite task to recognize Bengali Sign Language in real time.

Aryanie and Heryadi [22] have developed the American Sign Language based Finger spelling recognition system using k-Nearest Neighbour classifier method. The research problem is how fingers-pelling can be represented easily and how to increase the accuracy of the recognition. The research is characterized in two parts, one is finger pose feature is a vector of normalized color histogram and finger spelling recognizer is k-Nearest Neighbour. The dataset consists of around 500 plus samples of each letter from a-k excluding j. In the system the image patterns are represented through normalized color histogram and every image is represented by red, green and blue channels with a 16 bin histogram then it is merged into a 48-dimensional vector. To reduce data dimensions Principal Component Analysis is applied. The dataset is represented by the  $N \times m$  matrix where  $N$  is the number of samples and  $m$  is histogram bins. The researchers have used k-NN mainly as it is simplest in computer vision and it can differ computation of an unlabeled data. In this research, alphabet B got the highest accuracy with 83% following h, a, i, e, g, c. k, d, f. The researchers have also found that to represent finger pattern appropriately normalized color histogram is the right process. Moreover, the k-NN classifier has its own limitations when the dataset becomes too large.

Hasan et al.[23] has developed Hand Sign Language Recognition for Bangla Alphabet which is to get rid of the communication hurdle between a person without hearing disability and a person having hearing disability. To implement this, they choose an Artificial Neural Network based approach. All the necessary features are extracted from the hand sign region of interest using Freeman Chain Code Approach which is used to represent a boundary using a straight line segment's connected sequence. For the recognition system, a total of 20 out of 50 bangla alphabets are used as the dataset. This whole system is designed based on three major phases. The initial phase includes image data extraction from the hand sign region of interest. This phase contains five sub phases. First, pre-processing is done from the image dataset and gets converted into RGB formatted pic. Secondly, the RGB formatted pic is converted into YCbCr formatted pic, which is defined as linear luminance color space. After metamorphosis to the YCbCr color space, both chromaticities of blue and red color's threshold values are used to represent the color of skin. Fourthly, they are converted into a binary (black and white) formatted image which is used to obtain the shape of a hand sign and they are filled using some morphological

operations to process them in an appropriate manner. Finally, the edges of binary formatted pictures of the hand sign rate of interest are extracted. This step is done using a system named Canny edge detector. After the successful completion of the first phase, feature extraction is initiated to identify the single hand sign. For extracting the features, Freeman Chain Code approach is used. In the final phase, hand signs are recognized using the Artificial Neural Network containing multiple layers. For this, a backpropagation method is used and the network is trained with a huge number of inputs and outputs. For the implementation of the whole process, various input patterns of hand sign are used. After the complete implementation of the whole system, the accuracy is calculated as 96.5%.

Hoque et al.[24] has developed a model to identify BdSL using R-CNN where they developed their own dataset Bangladeshi Sign language Image Dataset (BdSLIm-set) to train their system. They have taken their train and test set using various backgrounds. Their system locates the position of the gesture for detection. Some of their pre-processing gestures are - morphological operation, color based segmentation etc. The system's main goal is to be fetched into the CNN with different backgrounds and lighting conditions so that the author's preprocessing stays minimal and more accurate. Three colors, Three boxes of height and width ratios are represented for the CNN to focus on the Reason of Interest (RoI). The respective sizes are - 128\*128, 256\*256, 512\*512 and the height-width ratios are 1:1, 2:1 and 1:2. The main algorithm that is used is Faster R-CNN where CNN generates a feature map and a network Regional Proposal Network (RPN) proposes regions with high probability of containing the desired object. In the network architecture RoI pooling layer is applied and the inputs are reshaped into a fixed size to feed into a fully connected layer. From the RoI feature map the softmax layer predicts the proposed region. In their system, an input image is divided into several anchors in CNN feature maps. RPN receives the reference boxes from the feature map and generates a good set of proposals of being objects. After the RPN stage, Max-Pooling is applied to every region. In order to identify Bengali sign language in real time, the authors applied a robust system and also the inputs are all kept under 200kb size limit and the resolution of 700\*1200. They only converted 10 different sign letters with a dataset of 100 pictures each. The train test set ratio is 8:2. After applying all these processes the output accuracy is about 98.2% with a loss of 0.07538 and the detection time was really fast of about 90.03 milliseconds. The limitation of their research is they could not apply it to all the Bengali Alphabets, some of the alphabets were not clearly recognized by the system but the authors could get the output in real time on the alphabets they applied on.

Another modern model was developed by Podder et al.[25] where the authors also created their own dataset BdSLHD-2300 in combination with the dataset BdSL-D1500. The authors classified Bangla sign language based on the usage of hands - one hand gestures or two hand gestures. In their first dataset BdSL-D1500 there are 87 classes of images - 38 one hand gestures, 36 two hand gestures, 10 digits, 2 numerals, 1 counting gesture. Total number of images in the dataset is 132,061 and the images that are collected from videos are RGB color images. The authors created a dataset BdSLHD-2300 which they used for training The image resolution was remained 331\*331 for both datasets. For ImageNet classification, three pre-trained

CNN models are used which are ResNet18, MobileNet\_V2 and EfficientNet\_B1 and for semantic segmentation of hand region and background removal, three CNN models such as DenseNet201 Feature Pyramid Network (FPN), U-Net, M-UNet are used. ResNet18 is mainly used in terms of vanishing gradients and decreasing accuracy after saturation. MobileNet\_V2 is the replacement of an expensive convolutional network which is usable in mobile devices and removes nonlinearities in narrow layers for smooth usage. The purpose of EfficientNet is Grid Search, U-Net is used for localization, MUNet is one kind of UNet with a skip connection. The datasets are arranged in a ratio of 70% training, 10% validation and 20% testing, stochastic gradient descent is used as an optimizer with a learning rate of 0.001 and batch size of 16. By applying all these methods the author managed to get 99% accuracy in their output images on an overall basis.

Fahmid et al.[26] has proposed another system which interprets Bangla sign language using image processing. This system has been completed in two different phases such as at first primary dataset has been created and for this fifteen classes have been prepared where there were almost 54 to 56 images each class. So a total of 830 figures have been used in this system for training and testing purposes. After collecting the dataset, a skin detection algorithm is applied on the images for detecting skin color pixels as 30 different people have participated in preparing the data. Here the skin detected image has been converted into 400 X 400 pixels. The YCBCR method has been used to detect the pixels and then it is converted into a binary image. This binary image is extracted with some noises for which a largest blob is picked out and other noises are subtracted and then only the main hand image is left. After that for feature extraction, a Bag of features method has been applied which is divided into four steps. First of all interest points have been selected, then SURF descriptors are calculated and then these values are stored into a feature vector. At last k-means clustering has been used for reducing the feature and computing the visual words. In this clustering process the feature vector is computed then randomly centroid is identified and visual words are recognized. Support vector machine(SVM) has been used for training dataset. 30% of total dataset means almost 16 figures from each section has been used for training purposes and the 70% means 38n figures from each section has been used as testing purposes. After applying SVM, the train set is compared with this trained classifier and almost 95% accuracy has been achieved. Even for some classes 100% has been matched successfully and for some classes it was 88%. On the other hand when the train set is matched with the test set the average 86% has been accurate as here for some classes the rate is quite low but for other classes the rate is almost the same. The author has been working on 15 Bangla sign letters because of the scarcity of assets and used static pictures but in the near future they will try to overcome the shortcomings.

Another system has been proposed by Javed et al.[27] which identifies the Bangla numerical sign language using CNN. The data has been captured through a webcam and this dataset contains a total 310 images where there are total 10 sign classes and for each sign there are 31 pictures. Then bandlet transformation is applied so that image structure and outline can be identified. After that logarithmic transformation has been done to fix the light effects and D-LBP calculates the dimension of image as well as fix the low resolution image. Then calculating the dimensions

of image skin color segmentation is done which converts RGB to YCbCr . Then it is converted into a binary image and the largest N blob is selected which extracts the main image of the sign, thus the qualified dataset has been prepared using alex net classifier which shows the desired output. For preparing this qualified data 750 figures have been taken and for testing purposes there have been 250 images with 0.0001 learn rate, max 20.0 epochs and min 64 batch size. This system has achieved 99.8% accuracy which has been calculated using precision, recall, true negative rate and lastly the accuracy rate using training and testing data.

# Chapter 4

## Proposed System

The system has been followed here is the Model Convolutional Neural Network with 6 Convolutional Layers, 3 Dropout layers, 2 Dense layers and Flattened once. Convolutional Neural Network consists of multiple hidden layers. The number of parameters grows rapidly with the increment of layers and tuning all of these may turn out to be a huge task. The time taken for tuning all the parameters are lessened by CNN. As we are working with Images, the raw data behind images are pixel values which are presented through a matrix. So the calculation with Images is already a tough thing. In CNN, dimensionality is reduced by using a sliding window with a size less than the input matrix. Kernel is a filter that is used to extract the features from the images, the kernel matrix moves over the input matrix following the strides. Strides are the value the matrix would move after each step. Kernel performs the dot product with the sub region of the input data and then gets the output matrix of the dot product. In each Convolutional Layer, Kernel Size = 4 has been taken. In this model, 2 convolutional layers and a dropout layer have been added and it has been done for like 3 straight times before the last dropout layer. A pattern has been followed like 2 Convolutional Layers with Stride=1 and Stride=2 in the next one. Thus the layer captures the value of the same kernel value matrix with two different strides and with the dropout layer, it prevents overfitting. The Dropout layer usually randomly sets the frequency 0 with a frequency rate of each step during the training time and where the inputs are unchanged it sets  $1/(1\text{-rate})$  which calculates the best matching of the convolutional layers. The Number of Convolutional layers are 64 and 128 then 256. After 3 sets of Convolutional Layers and Dropouts, a Flatten layer has been used.

$$\textit{Flattenlayer} = 1 - \textit{DimensionalArray} \quad (4.1)$$

The flatten layer has been used to change the shape of the data from a vector of 2D matrices into the correct format for the upcoming layer which is Dense Layer. Then the Dense layer sends all the output from the previous layer to all its neurons and each neuron produces one output to the next layer.

$$\textit{DenseLayerOutput} = \textit{Activation}(\textit{Dot}(\textit{input}, \textit{kernel}) + \textit{Bias}) \quad (4.2)$$

With the help of it, the previous data can be fetched and the network becomes fully connected. ReLU activation layers have been used in the convolutional layers. ReLU helps to prevent the exponential growth in the computation required to operate the neural network. But in the dense layer, the Softmax function has been used because Softmax helps to learn complex patterns in the data. During Compilation, the Adam optimizer has been applied. The Adam optimizer helps to read and reduce all the noisy problems, can take the advantages of the sparse feature and obtain a faster convergence rate. However, for all of these useful features, the following system and following patterns have been used for the system.

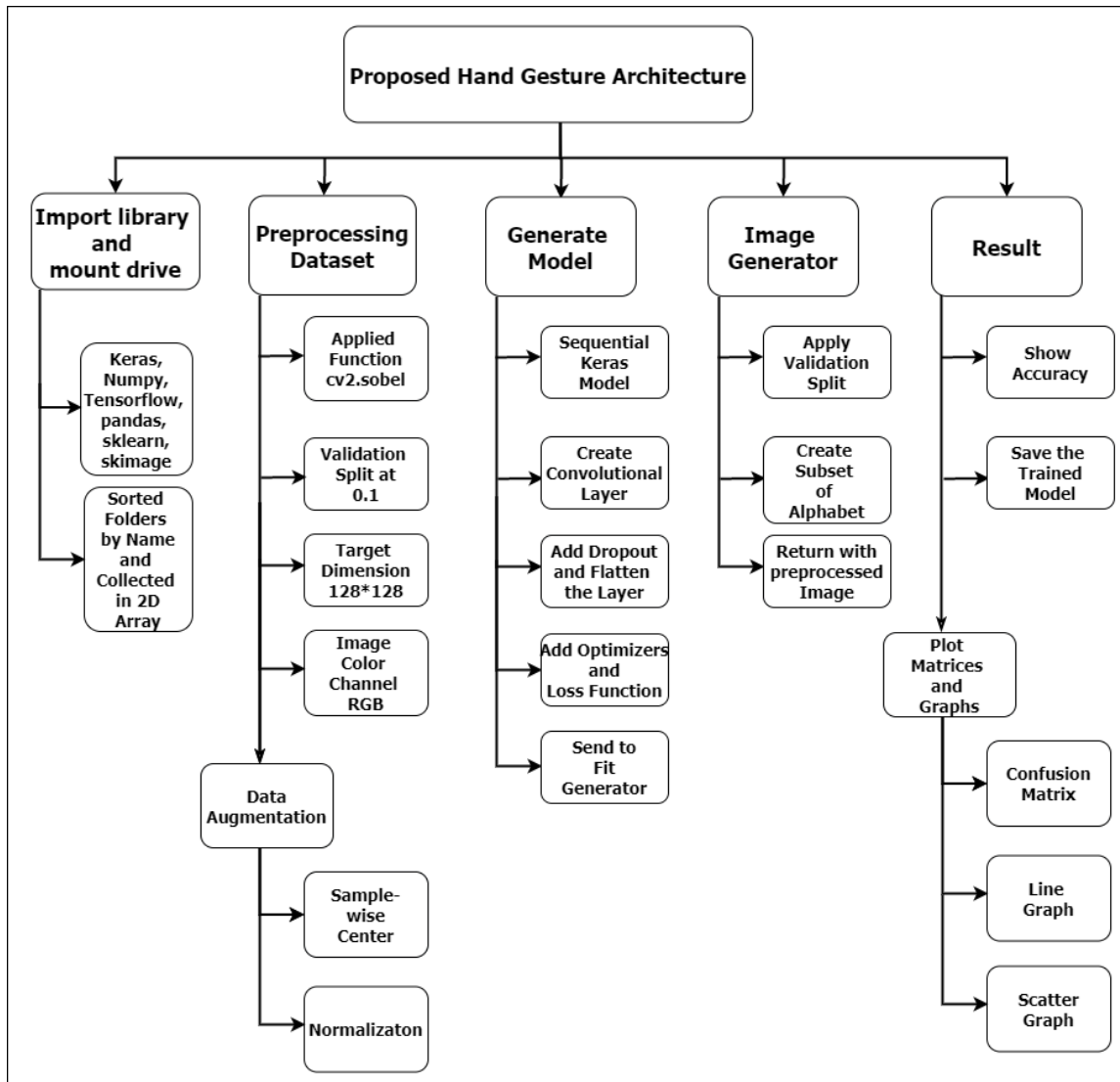


Figure 4.1: The proposed hand gesture recognition architecture

Hand gesture recognition is a difficult project to execute. That is why the whole project is divided into the following parts: (shown in fig. 4.1)

- Import Library and Mount Drive
- Preprocessing
- Generate Model



- Image Generator
- Result

## 4.1 Import Library and Mount Drive

To make the model, Python programming language has been used as it has many open source libraries which are needed to make a CNN-based model. In this project, Keras, NumPy, TensorFlow, sklearn, pandas, and skimage also have been applied. Keras is a useful open-source software library for artificial neural networks with a Python interface. The TensorFlow library is accessed through Keras. It is made to help in experimenting with deep neural networks quickly. It emphasizes usability, modularity, and extensibility. NumPy is a Python package that allows interacting with arrays. It also provides functions for working with matrices, Fourier transforms, and linear algebra. The lists in Python that act like arrays, however, are slow to process. NumPy intends to deliver a 50-fold quicker array object than ordinary Python lists. The array object in NumPy is named ndarray, and it comes with a number of helper methods that make working with it a breeze. Moreover, to make the computation faster, NumPy is being used here. TensorFlow is an open-source library for large-scale machine learning and numerical computing. TensorFlow combines a variety of machine learning and deep learning models and algorithms into a single metaphor that makes them practical. In Python, sklearn is the most useful and robust machine learning library. It uses a Python consistency interface to deliver a set of efficient machine learning and statistical modeling capabilities, such as classification, regression, clustering, and dimensionality reduction. Pandas is an open-source Python library. Pandas is a data analysis tool. After successfully importing all of these libraries, the folder is sorted name-wise in google drive and mounted.

## 4.2 Preprocessing

The next step is preprocessing the dataset. The primary task is to collect the ASL dataset (fig. 4.2) and build the own BdSL dataset (fig. 4.3) where there will be multiple pictures of each letter of Bangladeshi sign language using both white background and random real-world background.

After the dataset is being collected, the preprocessing part will begin. First of all, Sobel will be applied. The Sobel Operator produces a picture with the detected edges bright against a black background. Then a validation split of 0.01 has been used. This is an examined value and at this value, the best result for the project is achieved. The dataset has very high-resolution images. So all the images have been converted into 128\*128 dimensions. In this proposed system, RGB images have been worked with. Then the data augmentation part will be done. Data augmentation is a set of techniques for producing additional data points from existing data to artificially enhance the amount of data. Making modest adjustments to data or utilizing deep learning models to produce additional data points are examples of this. By creating fresh and varied instances to train datasets, data augmentation can help improve the performance and results of machine learning models. A machine learn-

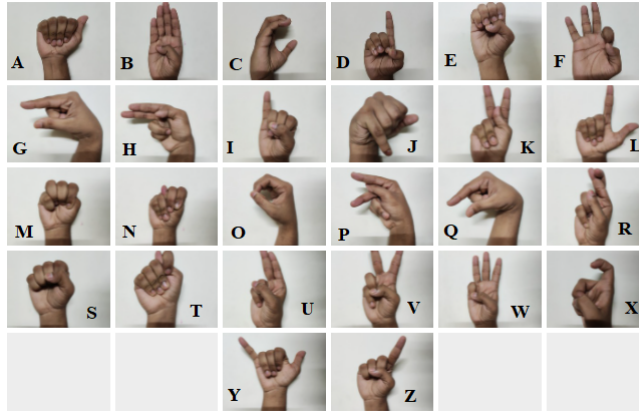


Figure 4.2: ASL established English Alphabets.



Figure 4.3: BdSL established Bangla Alphabets.

ing model works better and more correctly when the dataset is rich and sufficient.

In fig. 4.4 and fig. 4.5, the preprocessed train set for BdSL and ASL is shown respectively.

For training and testing purposes, for BdSL, 150 pictures have been taken of each alphabet in both Bangla as well as English with small changes in each picture for the algorithm to learn the pattern of the picture (shown in fig. 4.4). Among these 150 pictures, 100 of these from each alphabet are taken in front of a white background and 50 of these from each alphabet are taken in front of a random background for the Train set. 15 images of each alphabet have been shifted to the test set and for validation 0.1 or 10% of the overall images will be used for validation.

For ASL, the global ASL dataset has been collected that is usually used for American

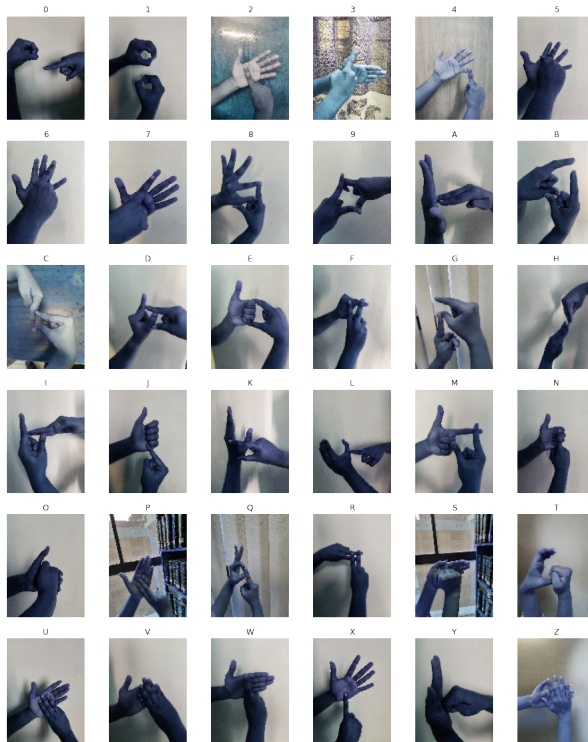


Figure 4.4: Train set for BdSL.

Sign Languages (shown in fig. 4.5). The dataset has 87000 images in total for 29 classes, 3000 pictures for each alphabets where 300 of them are used for test sets and validation split has been kept to 0.1. 26 English Alphabets and 3 extra for (delete, nothing and space) in real-world usage.

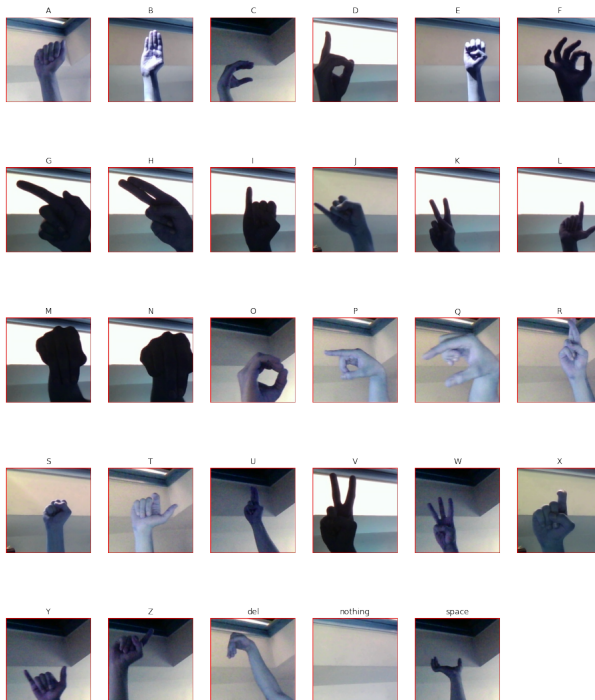


Figure 4.5: Train set for ASL.

### 4.3 Generate Model

After the preprocessing part, now the dataset is ready to make the model. The model is built using the Convolution Neural Network. The model is built in three phases mainly. In the first phase, CNN image classifications need an image as an input (fig. 4.6). An input image is seen by computers as an array of pixels, depending on the image resolution. It will see a matrix (height \* width \* dimension) based on the image resolution.

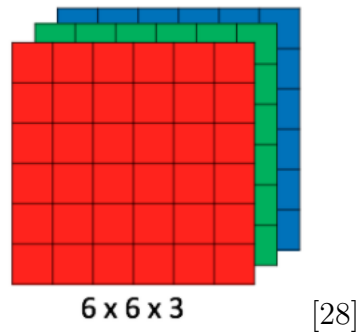
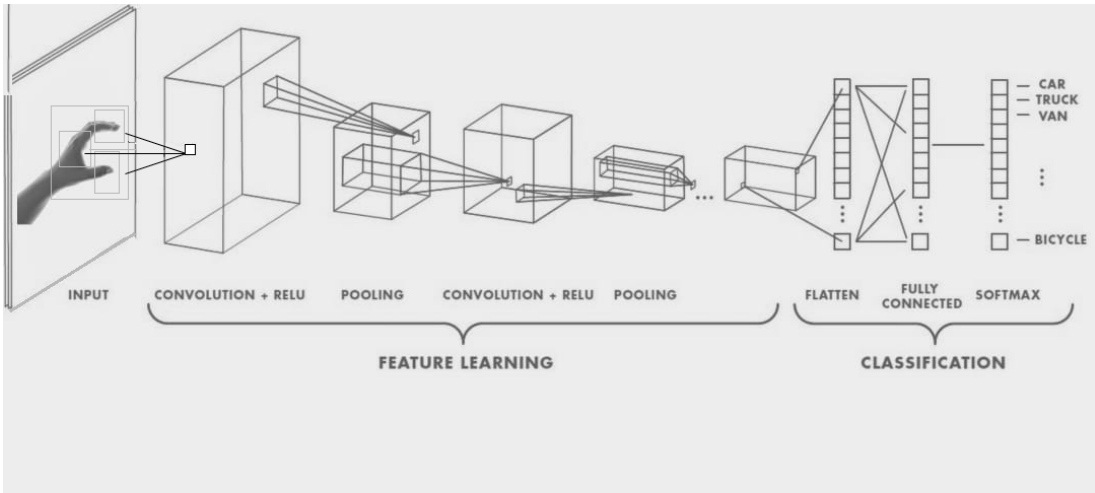


Figure 4.6: Array of RGB Matrix.

In fig. 4.6, it is shown that an image with the height=6, width=6, and dimension=3 is stored in three dimensional matrices. Here dimension 3 refers to an RGB image.

The input is processed in the next phase of the CNN. At the end of this step, the Feature Learning process is completed. In fig. 4.7, the whole process is illustrated.

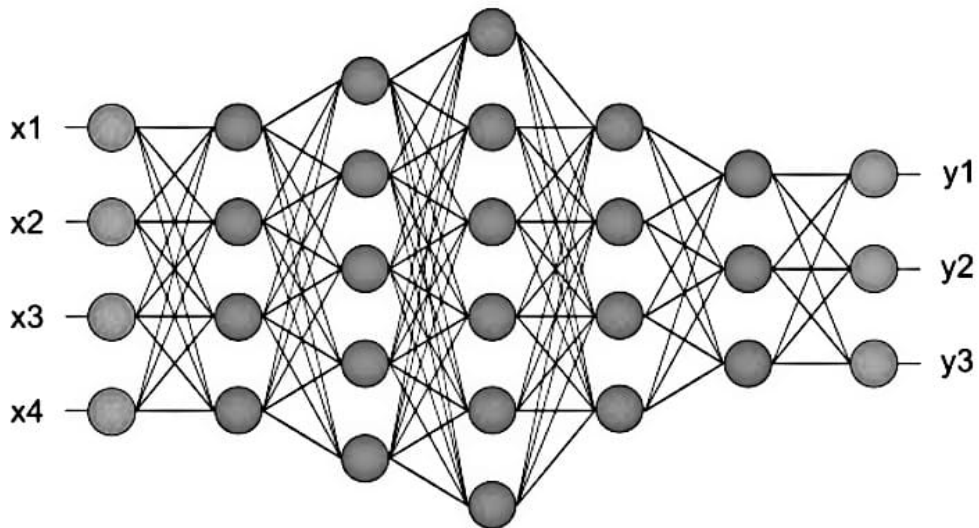
Here (fig. 4.7), the hand gesture image is the input and it is sent through a succession of convolution layers using filters (kernels), fully connected layers, Pooling, and the softmax function to identify an object with statistical values within 0 and 1. In the first layer, extracting features from an input image is convolution. By learning geometric features with small squares of input data, convolution preserves the link between pixels. It's a mathematical process with two inputs: an image matrix and a filter or kernel. After multiplying two of these matrices a feature map is got. Thus the convolution of an input gesture can identify the edge of hands, and fingers. Besides, it can make the gesture blur or sharpen. All of these depend on the type of kernels. After that stride operation takes place. The number of pixels that shift over the input matrix is called the stride. Then the filters are moved one pixel at a time when the stride is 1. Again the filters are shifted two pixels at a time when the stride is two, and so on. In the following step, padding is needed based on necessity. It is examined that in many cases the input image does not fit. So padding helps to get rid of this problem. In padding, either the pictures are padded with zero or the area of the image is removed where the filter didn't work. This is known as valid padding, as it maintains only the image's legitimate parts. In the next part, ReLU will be applied (fig. 4.7). The goal of ReLU is to add non-linearity to ConvNet. Because the data which is to be learned by ConvNet in the actual world is non-negative linear numbers. Following that, the model passes through the pooling layer.



[28]

Figure 4.7: A Neural network system taking hand gesture and passing through many convolutional layers.

When the photos are too huge, the pooling layers portion would lower the number of parameters. Spatial pooling, also known as sub-sampling or down-sampling, decreases the dimensionality of each map while preserving crucial data. There are several types of spatial pooling: Average Pooling, Maximum Pooling, sum pooling, etc. The largest element from the corrected feature map is used in max pooling. The average pooling could be taken from the largest element. Sum pooling is the sum of all elements in a feature map. In the last phase of CNN, the processed data is sent for classification under various classes.(fig. 4.8)



[28]

Figure 4.8: After pooling layer, flattened as FC layer.

The matrices have been flattened into a vector and have been fed into a fully connected layer. The feature map matrix will be converted to a vector. These features have been integrated to generate a model using the fully connected layers. Finally, an encoder like softmax and sigmoid is used to classify the outputs as क,ख,ग for BdSL and A,B,C for ASL. (shown in fig. 4.9)

The summary of generating the model is shown below(fig. 4.9).

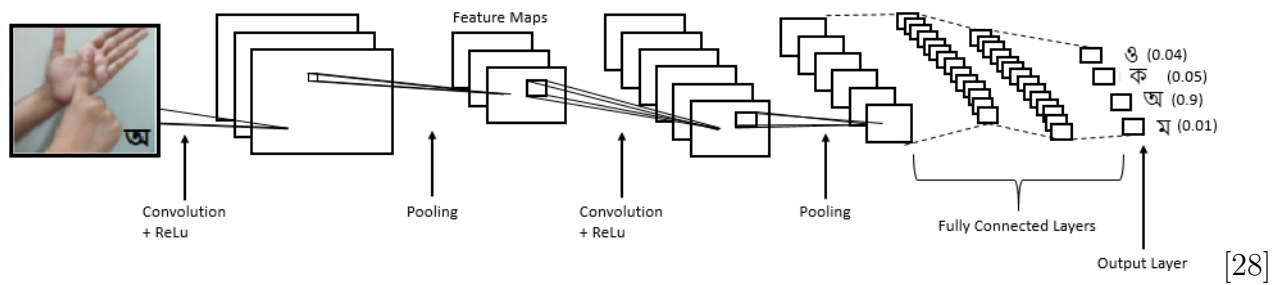


Figure 4.9: Complete process of Image recognition using CNN.

## 4.4 Image Generator

Once the model is ready, the validation split will be applied again. A subset of alphabets will be created and randomly an image will be generated. Now it will be cross-checked with the preprocessed model.

## 4.5 Result

Finally, in the result section, the accuracy is calculated. After getting a good accuracy, the trained model has been saved. To understand more easily, the accuracy has been plotted in the Confusion matrix and the comparison between accuracy vs loss also has been plotted in the Line graph and Scatter graph.

# Chapter 5

## Platforms, Language and Libraries

### 5.1 Programming language

A programming language is a set of sequential commands, instructions and syntax in order to perform some specific tasks by computer or any computing device. Programming languages that are used by the users are referred as High Level Language. On the other hand, when the programming language gets translated through compiler or any translation medium in such a way that the machine can understand, that becomes Low Level Language.

#### **Python**

Python is one kind of high-level language. Recently, this language has become well known due to its versatile usability, powerful libraries and simplicity. Python was first introduced in 1991 by Python Software Foundation. Nowadays it is widely used in the important sectors like software and website development, data analysis, machine learning, blockchain and so on. Due to the convenience of dataset training and testing, data visualization, analyzing and ease of understanding, Python has been used as the programming language of the proposed system.

### 5.2 Platforms

For python coding implementation of the proposed system, one integrated development environment (IDE) and one notebook based code editor has been used. The names of the two platforms are Google Colab and PyCharm.

#### **Google Colab**

Google Colab is a popular notebook based code editor of Google Research. Google Colab does not need any installation or download to run the code. Rather it allows multiple persons of a team to write down code, modify code if needed and execute the code using browser. As a consequence, working on the same page is easier in google colab than any other platform as it reduces the problems that arises using different computers or laptops. Moreover, it is very helpful for the implementation of any deep learning and data analysis related python programs. Google Colab contains many powerful libraries, which omit the necessity of the installation of several powerful libraries and lessen workload. Therefore, Google Colab has been applied

in the thesis.

### **PyCharm**

Although Google Colab resolves almost all the problems of collaboration of the code, it has a few limitations too. Among them, the most dominant limitation is the space limit. Although, all the datasets that are taken as input and trained were compressed at the end, it was not possible to fix the problem of space limit while datasets were not compressed. Meanwhile, PyCharm can resolve this problem. In addition, using PyCharm helps to fix the codes very easily due to its smart code completion and quick fix system. This system not only compiles the code and identify where the error or problem is but also provides some possible solutions to the problem.

## **5.3 Libraries and Modules**

Python is a programming language that requires a handful number of libraries and modules intending to produce productive outcomes. Likewise, for ensuring proper implementation of the proposed system, usages of some libraries and modules were required.

### **os**

The os is a standard module of python programming language. It is primarily used for fetching contents from folders, creating and changing directory. In this proposed system, OS was used in order to fetch necessary contents (for example: dataset) from folders which are located in google drive.

### **matplotlib**

The matplotlib is a plotting library for the Python language. It is used for mainly data visualization purpose through several static and interactive Mathematics based diagrams. This was one of the most important library in the proposed system. Sign language containing images, graphs and charts describing accuracy, loss, epoch and predicting and showing output- all were done by this library.

### **skimage**

The skimage (sci-kit image) is an open source package for the python programming language that is typically used for image pre-processing. It provides several useful and productive functionalities related to machine learning and statistical model including segmentation, regression, filtration, feature detection. For the proposed system, skimage library has been used for the sobel filter process and scharr transformation. Sobel edge detection is a procedure which is chiefly based on gradient in order to figure out significant changes in the first derivative of an image using a couple of square convolution masks of order 3 for the estimated gradient in the x and y directions. Scharr transform shows the resulting magnitude of the gradient edges.

### **cv2**

Image processing and computer vision related problems are solved using cv2 mod-



ule of the OpenCV library. For the implemented system, cv2 module has been used for the execution of the Sobel operator in order to pre-process the images that are used in dataset by calculating the composite gradient towards both x and y-axis.

### **seaborn**

The seaborn is a library which is used for data visualization like Mathplotlib which shows clearer and more informative visualization by having high-level interface. In this thesis, Seaborn is mainly used for showing detailed pictorial description of confusion matrix.

### **pathlib**

The pathlib is a standard module for python programming language. It creates a readable and convenient procedure for building up paths by turning addresses into strings.

### **keras**

The keras is a powerful open source library for python programming language. It is implemented for the development and the evaluation of the deep learning models. For the thesis, keras has been deployed in order to build neural network layers by taking input for doing some computational analysis and resulting output, doing the pre-processing activities, loading and fitting the datasets.

### **tensorflow**

The tensorflow is a popular open source library for making the computational process faster of machine learning. It supports various machine learning algorithms and data visualization processes. For the proposed system, random seeds has been set in order to make the TensorFlow code reproducible.

### **pandas**

The pandas (Python Data Analysis Library) is a popular open source library for data science and machine learning. It is generally used for steering real life raw data and giving them cogency. The proposed system required pandas for converting and extracting the raw model weight matrix into cleaned and transformed dataframe.

### **numpy**

The numpy (Numerical Python) is a effective data structure based library which is capable of doing a wide range of mathematical calculations between multidimensional array objects. For the proposed system, numpy library has been used in order to do the calculations in the confusion matrix and ceiling the row values of the function that was used for sample plotting.

# Chapter 6

## Dataset

To have a favorable implementation for the system a dataset has been created for “Bangla Sign Language”. The dataset is important for the increment of the effectiveness of the system. As our main goal is to develop a system which can work in real time so a large dataset is necessary.

Bangla Sign Language includes 36 letters. The dataset consists of 100 pictures of each letter which creates a dataset consisting of 3600 pictures. The pictures have been taken in five different angles- Front, Slightly Right, Slightly Left, Slightly Top and Down. This variation has been kept because the system is for real time and in real time the user won't always give a front to front input. For every letter there are at least two different people's hands so that there remains a different color contrast. The pictures have been taken by keeping white background. The average size of every photo is 3.5-4.5 megabyte. The size differs mainly because of framing angles. The lighting of every picture has been kept the same. The total size of the dataset is 13.3 gigabytes which include 36 folders and every folder represents every letter of Bengali sign language. Different persons' hand gestures have been used to create the dataset.

To train the dataset, every picture has been converted into 128\*128 pixels. This conversion has been done due to the size of the image the train programme has been over fitting. But after the conversion in the Pycharm environment it takes 15minutes to train the dataset. For American Sign Language, 64\*64 has been used but for BdSL, 128\*128 has been used just because a picture with a better dataset is needed to be used. For example, by looking at the three pictures (fig. 6.1) given below, there are similarities among them.

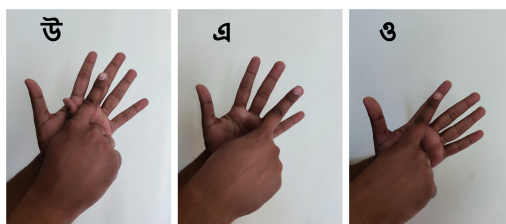


Figure 6.1: উ, এ, ও

The position of the left hand is the same in three of the letters. Even the gesture of the right hand is almost the same. The position of the index finger of the right hand is the differentiating factor in these 3 letters. In the left picture the right hand's index finger is on the left hand's middle finger. In the middle picture the right hand's index finger is on the left hand's index finger. In the right picture the right hand's index finger is on the left hand's ring finger. So for this reason the more accurate picture will give us a more accurate train test result. But in the case of ASL, there is no usage of two hand gestures. That's why to train the dataset more accurately the conversion has been done with higher pixel value. Manually generated codes have been used to train the dataset. As Bengali alphabets are not acceptable as a string value, a code value has been given to all the alphabets. These codes are all English alphabets. The codes of each letter is given below: (fig. 6.2)

অ	আ	ই	উ	এ	ও	ক	খ	গ	ঘ	চ	ছ	জ	ঝ	ট	ঠ	ড	ঢ
2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	K

ত	থ	দ	ধ	ন	প	ফ	ব	ভ	ম	র	ল	স	হ	য়	ড়	ং	ঃ
L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	V	J	0	1

Figure 6.2: String value used for BdSL

The Bengali letters are given as string values under these coded variables and give output on the basis of that.

While creating the database it has been acknowledged that the images in the dataset must be clear so that the algorithm can read the pictures. Any kind of object in the frame has been avoided. Only the hand and white background is seen. Unfortunately the shadow of the hands can be seen as there wasn't any way possible to remove but the shadow didn't create any problem while training the dataset. Different people's hand has been used to create the dataset to keep a versatility in the color contrast of hand and five different angles have been used as there is a high chance that the input is not straight in frame. Sometimes the input will be angular. With the help of angular pictures, the efficiency will be higher in these cases.

BdSL are two types. One is using just one hand and another is using both the hands. We have worked on sign language where both hands are used as it is the most convenient while communicating between each other. One hand communication is not preferable. The dataset also has been created one the basis of two hand BdSL. Every letter has been represented through 100 pictures. The description of each alphabet is now being discussed.

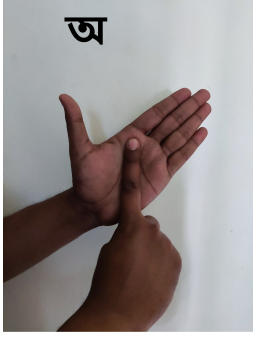


Figure 6.3: অ



Figure 6.4: আ



Figure 6.5: ই



Figure 6.6: উ



Figure 6.7: এ



Figure 6.8: ও



Figure 6.9: ক

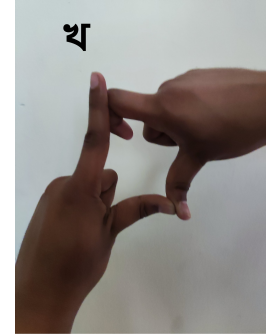


Figure 6.10: খ



Figure 6.11: গ

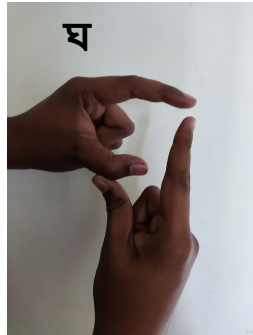


Figure 6.12: ঘ

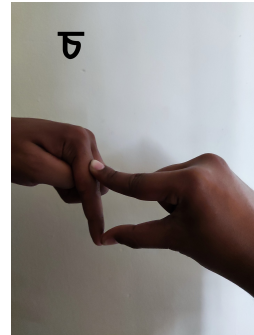


Figure 6.13: চ



Figure 6.14: ছ

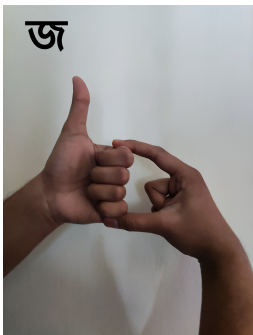


Figure 6.15: জ

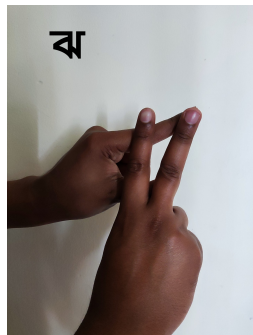


Figure 6.16: ঝ

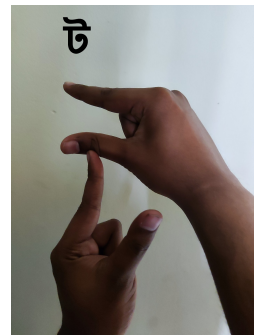


Figure 6.17: ট

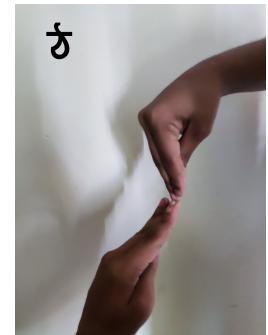


Figure 6.18: ঠ

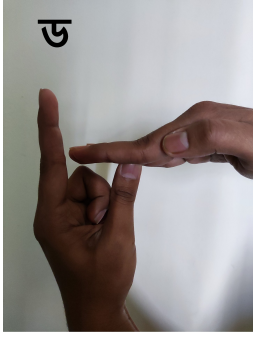


Figure 6.19: ড

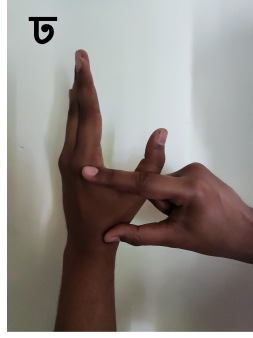


Figure 6.20: ঢ

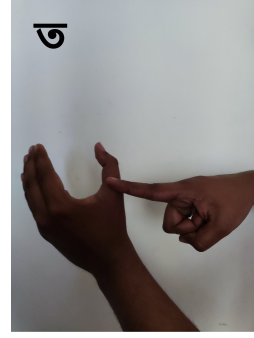


Figure 6.21: ত

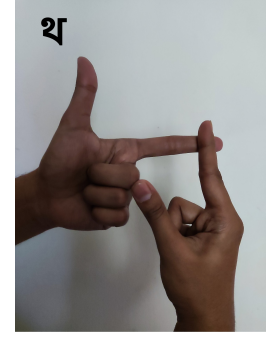


Figure 6.22: থ

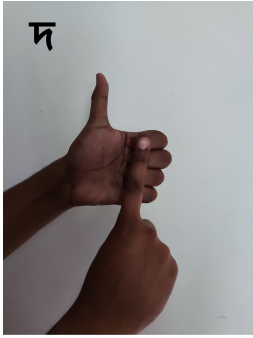


Figure 6.23: দ

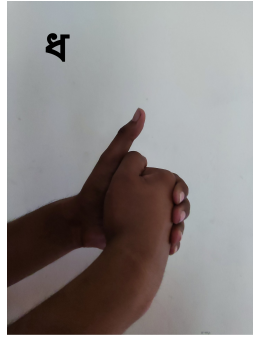


Figure 6.24: ধ



Figure 6.25: ন

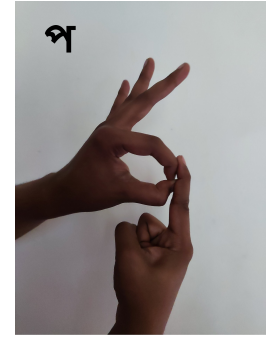


Figure 6.26: প

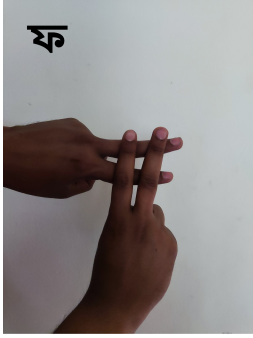


Figure 6.27: ফ

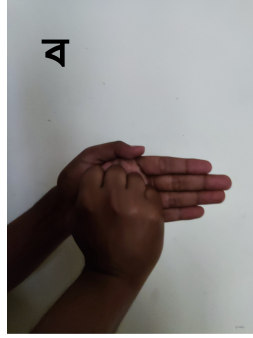


Figure 6.28: ব

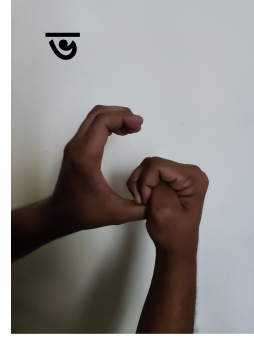


Figure 6.29: ভ

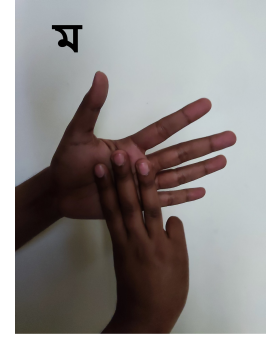


Figure 6.30: ম

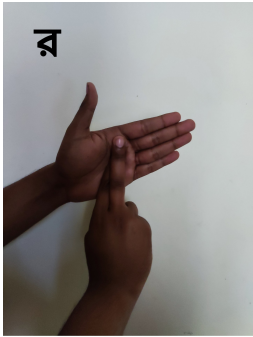


Figure 6.31: র



Figure 6.32: ল



Figure 6.33: স



Figure 6.34: হ



Figure 6.35: য

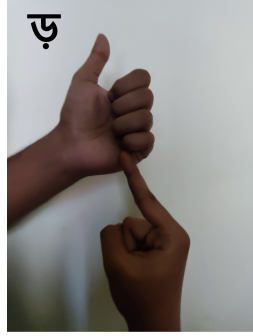


Figure 6.36: ড

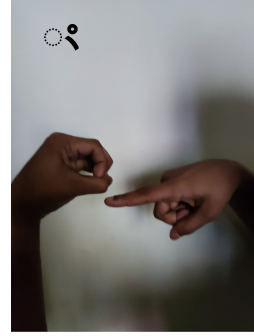


Figure 6.37: ণ

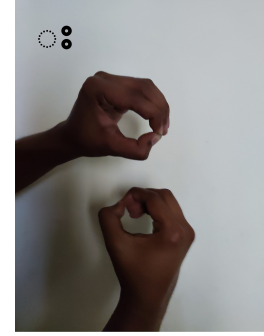


Figure 6.38: ঃ

These figures (7.3-7.38) represent BdSL of each alphabet.

**অ:** This letter is represented through the right hand's index finger positioned at the left hand's palm. (fig. 6.3)

**আ:** Right hand's index finger is pointed at the left hand's thumb. (fig. 6.4)

**ই:** All fingers of the left hand are open widely. The index finger of the right hand is pointing at the little finger of the left hand. (fig. 6.5)

**উ:** Fingers of the left hand are widely opened. The index finger of the right hand is pointed at the middle of the middle finger of the left hand. The thumb of the right hand is slightly opened and situation at the palm of the left hand. (fig. 6.6)

**এ:** The fingers of the left hand are widely opened. The index finger of the right hand is pointed at the ring finger of the right hand. (fig. 6.7)

**ও:** This time as well the fingers of the left hand are wide open. The index finger of the left hand is pointing at the middle of the index finger of the right hand. (fig. 6.8)

**ক:** The thumb and index finger of the left hand touches the index finger and the thumb of the right hand as it creates two rectangular shaped boxes. The folded middle finger of the right hand is seen through the lower rectangle. (fig. 6.9)

**খ:** The thumb and index finger of the left hand stands like the English letter "L". The right hand's index finger is folded and positioned at the middle range of the left hand's index finger. The thumb of the left finger is pointed at the head of the thumb of the right finger. (fig. 6.10)

**গ:** The left hand is standing vertically with a wrist angle of 180 degree. The palm of the left hand is facing the right side. The left hand's four fingers without thumb are horizontally pointing at the palm of the left hand. The thumb of my right hand is folded. (fig. 6.11)

**ঘ:** The thumb and index finger of the right hand stands like the English letter "U". The thumb and the index finger of the left hand also like "U" but horizontally stands

on the thumb of the right hand. There is a slight gap between the index fingers of both hands. (fig. 6.12)

൮: The index finger of the left hand is pointing downward. The index finger of the right hand is touching the lower part of the left hand's index finger and the thumb of the right hand is touching the top of the index finger of the right hand. (fig. 6.13)

൹: The left hand's thumb and index finger create a "U" shape. The right hand's thumb and index finger also creates a "U" shape but that is horizontally and pointing towards the left and positioned on the top of the left hand's thumb. The right hand's index finger touches the middle part of the left hand's index finger. (fig. 6.14)

ൺ: The left hand is shaped like a thumbs up shape. The right hand's index finger and thumb touching the little finger and index finger, The middle finger of the right hand is folded. (fig. 6.15)

ൻ: The index finger of the left hand is pointing towards the right side and the right hand's index finger and middle finger creates a "V" shape on it. (fig. 6.16)

ർ: The index and thumb of the left hand is shaped like "U". The right hand's thumb and index are also shaped like "U" but they are horizontally pointing towards the right and thumb's middle thumb is on the top of the left hand's index finger. (fig. 6.17)

ൽ: The two hands remain 180 degrees vertically on top of another. The mid finger of both hands touches each other. (fig. 6.18)

ൾ: The left hand's index and thumb are again shaped like "U".the whole right hand is on top of the left hand's thumb, facing palm towards ground and there is a gap between the left hand's index and right hand. (fig. 6.19)

ൿ: The left hand is again created an "U" shape but now with all fingers. The right hand's index finger is placed in the gap between the right hand's thumb and index finger and the thumb is pointing at the wrist and other fingers of the right hand are folded. (fig. 6.20)

ൾ: Continually the left hand is making an "U" shape but this time one side is thumb and other side is the other four fingers. The right hand's index finger is pointing towards left side and touching the lower point of thumb of the right hand. (fig. 6.21)

ൿ: The left hand is shaped like an "L" with a thumb and index finger. The index finger is point towards the right side. The right hand is creating a "U" shape where the index finger is touching the left hand index finger's head. The middle finger of the right hand is folded but can be seen. (fig. 6.22)

ൿ: The left hand is shaped like the thumbs up symbol. The index finger of right hand is situated on all the folded fingers of the left hand. (fig. 6.23)

ॐ: The left hand is shaped like the thumbs up symbol but the four fingers here slightly folded. The right hand's all fingers are folded and positioned at the palm of the left hand. (fig. 6.24)

ॐ: The palm of the left hand is open and facing towards the front. The index finger and the middle finger of the right hand is situated at the palm of the left hand without any gap between the fingers. (fig. 6.25)

ॐ: The thumb and index finger of the left hand creates a circle and the other three fingers are standing straight. The index finger of the right hand is pointing upwards and touching the connection point of the thumb and index finger of the left hand. Other finger of the right hand is folded. (fig. 6.26)

ॐ: The index finger and the middle finger of both hands situated on each other making a symbol like hash(#). (fig. 6.27)

ॐ: The left hand fingers are open and situating horizontally without any gaps between the fingers. The fingers of the right hand are folded and positioned at the palm of the left hand. (fig. 6.28)

ॐ: The left hand fingers create a half circle and the circle is open towards the right side. The right hand's fingers are folded and take the left hand's thumb inside the fingers of the right hand. (fig. 6.29)

ॐ: The fingers of the left hand are widely opened and horizontally towards the right. The three fingers of the right hand, which are the ring, middle and index finger, are positioned at the palm of the left hand. (fig. 6.30)

ॐ: The left hand is horizontally to the right and there is no gap between the fingers though the thumb and the other fingers have an angle of 90 degree. The index finger is under the middle finger of the right hand and both pointing at the palm of the left hand. (fig. 6.31)

ॐ: The left hand's fingers are wide open and the palm of the left hand is facing front. The index finger of the right hand is at the lower part of the palm of the left hand. Other fingers of the right hand are folded. (fig. 6.32)

ॐ: The left hand is standing vertically facing the right side. The left hand's fingers are folded and the gaps of the first and second knuckle touch the palm of the left hand. (fig. 6.33)

ॐ: The five fingers of both hands crossed each other. The left hand is at back facing front and right hand is facing back is at front. (fig. 6.34)

ॐ: The left hand's four fingers, except the thumb, are half folded. Thumb is open. The right hand's four fingers are covering the left hand's four fingers. The thumb of the right hand can't be seen. (fig. 6.35)



☞: The index finger of the right hand is pointing upwards. The left hand is shaped like the thumbs up symbols and positioned at the top of the right hand's index finger. (fig. 6.36)

☞: The left hand's fingers create a circle. With the index finger the right hand is pointing at them. There is a slight gap between two hands. (fig. 6.37)

☞: The both hands fingers create one circle each. The left hand's circle is above the right hand's circle. (fig. 6.38)

# Chapter 7

## Methodology

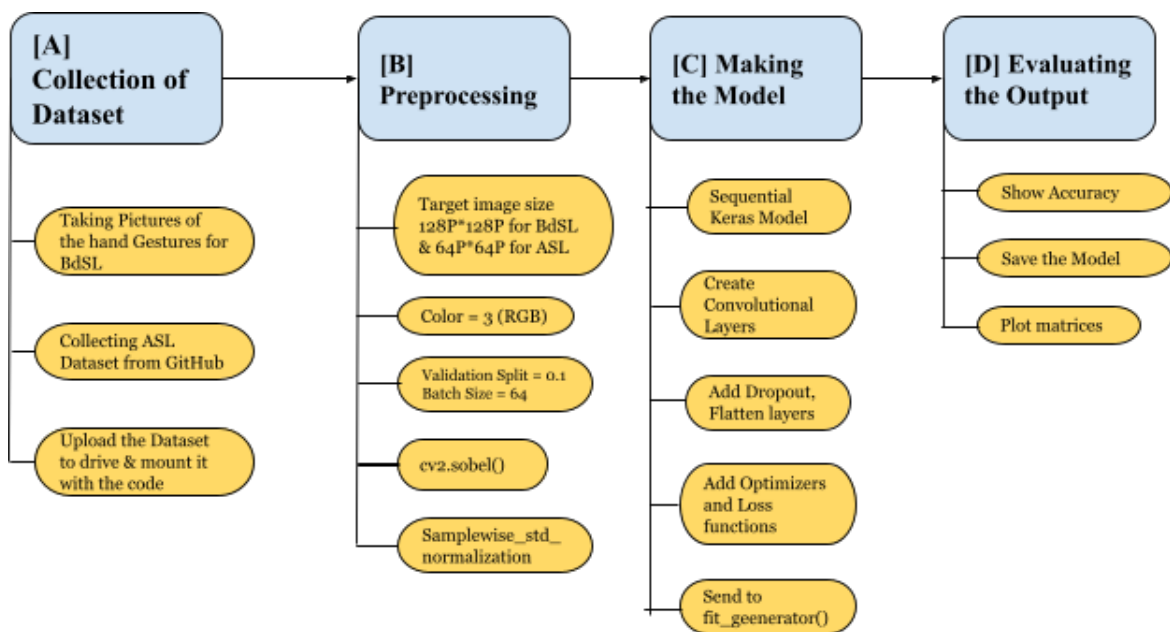


Figure 7.1: Work Methodology

In fig. 7.1 Detecting Hand gestures, Recognizing its values and checking the accuracy is sequentially described. Google Colab is the platform for this system. The best feature of Google colab is that it can import libraries without even downloading it. Time is a huge factor here as the dataset consists of more than 92500 images. Also, while working with a large dataset - one ASL global dataset and one BdSL (Two Hand Gesture) dataset made by the authors, the google colab can fetch the directories without even particularly uploading it. It just needs to exist in google drive.

Fig.7.1 describes how the proposed system will work. For English Alphabet recognition ASL Alphabet : Image Data set for alphabets in the American Sign language has been used.<sup>1</sup> For Bangladeshi Sign Language recognition a new dataset has been created using Xiaomi Poco F3, Realme 5 devices with 100\* (White background) and 50\* (Random background) for each Bengali Alphabet. The dataset contains both male and female hands regarding skin tones so that the accuracy versus test set

<sup>1</sup>GitHub Repository: <https://github.com/grassknotted/Unvoiced>

comes out even better. Then the Dataset is uploaded to Google Drive with separate folders and then mount that particular drive to the code with proper addressing. All the alphabets in both train and test set have their own folders for easy path finding. Some important libraries are needed to be imported in the code for this particular part. For instance, glob and cv2. Glob is the short form of Global which is used in the code to return all file paths that match a specific pattern like all the images in a folder of the same alphabets. Cv2 is the import name of OpenCV-python. In the initial stage, cv2 is used to read the images and show it as an output to ensure all the alphabets from the train and test folder have been read. So, the initial step is collection of the Dataset [A].

The next step is the Preprocessing Stage [B](shown in fig. 7.1). The ASL Dataset, all the 87000 pictures are 200\*200 pixels each and the pictures are 3000\*4000 pixels each in the BdSL dataset. The first step of preprocessing is setting the target dimension of the images to 128\*128 pixels. Color code RGB and Validation split to 0.1 which means 10% of the images of overall datasets will be used for validating the training and testing images. Batch Size is kept 64 means the number of training examples utilized in one iteration will be 64. The images are sent to a function called Sobel which computes an approximation of the gradient of an image intensity function. The sobel operator combines Gaussian Smoothing and differentiation. Then comes the part of making a generator. Augmentor options have been selected in the generator making method. Samplewise\_center is set to true as it takes each sample and normalizes its features such as the end it turns up being a unit vector. The dataset is then set for Standard Normalization. The inner calculation of it is subtracting the mean value of each feature and a division by the standard deviation. This way the features are set between 0 and 1 and it converges very fast. The augmentor options are set for both validation and preprocessor where all the initial trained set is addressed. When the system will not find any trained model in the particular directory, it will then start preprocessing the directory that has been fetched to the system. Otherwise, it will simply load the model that has been trained already and available in the directory.

```

model = Sequential()

model.add(Conv2D(64, kernel_size=4, strides=1, activation='relu', input_shape=target_dims))
model.add(Conv2D(64, kernel_size=4, strides=2, activation='relu'))
model.add(Dropout(0.5))
model.add(Conv2D(128, kernel_size=4, strides=1, activation='relu'))
model.add(Conv2D(128, kernel_size=4, strides=2, activation='relu'))
model.add(Dropout(0.5))
model.add(Conv2D(256, kernel_size=4, strides=1, activation='relu'))
model.add(Conv2D(256, kernel_size=4, strides=2, activation='relu'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dense(N_classes, activation='softmax'))

```

Figure 7.2: Snippet of sample code

When the preprocessing is done the next step is making the Model [C] (shown in fig. 7.2). Here Keras-Tensorflow is chosen because Keras is an open source neural network Application Programming Interface (API) for python. Keras is vastly useful for developing and evaluating deep learning models. The trained model is the

Sequential Model of Keras. Sequential Model API is a way of making deep learning models where an example of the Sequential Class is created and Model layers are created and added to it. Then the proposed system is created according to fig. 7.2. In the proposed system, there are 3 sets of CNN convolutional Layers with 2 convolutional sequential layers and a dropout in each step. Before the dropout function, 2 convolutional layers are similarly used with two different stride values. The stride values are kept between 1 and 2 which helps the fitting model to work on different matrices and make the best accuracy out of it. After each set of Convolutional hidden layers a dropout layer has been added. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time. Inputs not set to 0 are scaled up by  $1/(1 - \text{rate})$  such that the sum over all inputs is unchanged and provides with great accuracy and less chance for overfitting. Then the Flatten layer is added before the Dense layer at the end. Then a couple of Dense layers are added for receiving all the inputs from the previous layers and classify the image based on output from conv layers. Here, units have been taken 512 number of classes to neurons respectively. Units help to define the size of the output from a dense layer. Flattening in this system is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is sent as an input to the fully connected layer to classify the image. As a result, the spatial dimension of the input turns into a channel dimension. 3 different optimizers that have been used - adam, softmax and ReLU. ReLU activation function has been used in the convolutional hidden layers of the model. The rectified linear activation function that will output the input directly if it is positive and otherwise the output will be zero. So if the test is matched in hidden layers, the ReLU will show positive output, else the output will be negative. Softmax is used for generalization in the dense layer, after the convolutional layers. Softmax is here used as an activation function in the output layer that predicts a multinomial distribution function. During the model compilation, adam optimizer has been used. It is used to combine two gradient descent methodologies and accelerate the algorithm to take consideration of the exponentially weighted average value. The loss function that is used is the Categorical Cross Entropy Loss which outputs only one hot category encoding that values from 0 to 1. After building the model using required instructions, the model is set for fitting in the `fit_generator()` function.

The final stage is showing the output of the trained system [D] (fig. 7.1). Here 7 epochs have been run on the fitted model. 5 epochs have been tested first but the output seemed a bit less accurate. Again 8 epochs turn out to overfit the model. So 7 epochs are perfect for the preprocessed dataset. The system will then show output of the trained model. Here, `Getenv of Os` is used to extract the value of the environment variable key if it exists and provides an interface to interact with the operating system. The freshly trained model will be saved for further faster execution through this. Confusion Matrix, Accuracy, Loss, Scattered and Line graph will be plotted for visual representation of accuracy and loss.

# Chapter 8

## Result Evaluation

### 8.1 Result

In the table: 8.1, the accuracy of the system for both BdSL and ASL has been shown elaborately.

In BdSL, the system gives us an accuracy of 98.45%. The dataset is based on two handed communication of BdSL and has been created specially for this system. When the random background is added, till 5 epoch the accuracy was 98.05%. After running at 7 epoch the accuracy has increased to 98.45%. Meanwhile, in the white background (under regulated circumstance), the accuracy reaches at 99.07% after 5 epoch. In both cases, after running first epoch, the loss has been very high. However, from second epoch, the loss has started to come down drastically.

To work for ASL, The system has got an accuracy of 95.23%. Similar to BdSL, ASL experienced varied loss from initial epoch to rest of the epoch. The dataset is the global dataset of ASL.

Epoch	BdSL Accuracy - Real world Random Background (%)	BdSL Accuracy - White Background (%)	ASL Accuracy (%)	Loss - BdSL - Random Background	Loss - BdSL - White Background	ASL Loss
1	46.31	45.31	56.89	2.0219	2.0349	1.3935
2	90.47	92.72	88.34	0.3644	0.2537	0.3502
3	96.06	97.78	93.07	0.1321	0.0748	0.2085
4	97.33	97.59	94.35	0.0884	0.0740	0.1734
5	98.05	<b>99.07</b>	<b>95.23</b>	0.0691	0.0389	0.1488
6	98.22			0.0595		
7	<b>98.45</b>			0.0581		

Table 8.1: Result of the applied model

## 8.2 Comparative Analysis

Here in the table- 8.2 are some comparisons between some other works and the applied system. The comparisons have been done on the algorithm , dataset, optimizer, sign languages the system has worked on and mainly focusing on the accuracy.

Author/ Paper	Huda et al. [18]	Sanzidul et al. [11]	Olamiti et al. [17]	Ishara- bochon [16]	Kohsheen et al. [12]	Applied system
Dataset Collection	BdSL (3600 images)	BdSL (1800 images)	Global ASL	1000 BdSL Numeric	ASL (Sign to text 3000)	BdSL (5400), ASL (87000)
Pre- processing	Microsoft intelligent assistance, Microsoft voice command	128* 128 size	Hand gesture to text (Kinect sensor)	128* 128 size	OpenCV, Color RGB, 200*200 size	Color RGB, 128*128 size, augment- ation
Algorithm/ NN	ANN	ANN	Fast CNN, SURF, KNN	CNN	Linear SVM	Model Sequential CNN
Optimizer	ADAM, ReLU	ADAM, ReLU, Softmax	Unknown	ADAM	Gaussian filter	ADAM, ReLU, Softmax
Accuracy (%)	99	92.65	78	92.87	98.82	95.05(ASL), 98.45(BdSL)
BdSL (Yes/NO)	Yes	Yes	No	Yes	No	Yes
ASL (Yes/NO)	No	No	Yes	No	Yes	Yes

Table 8.2: Comparison between applied system and other systems

In terms of dataset, Sanzidul et al. [11] have used 1800 less images than the applied system. Moreover, Like Olamiti et al.[17] the system has used the global dataset of ASL(87000). Similar to the applied system, only Kohsheen et al.[12] have used color RGB images. As Sanzidul et al. [11] and Ishara-bochon [16] have worked with static images so both of them have taken 128\*128 pixel image sets. Furthermore, Ishara-bochon [16] has worked with CNN similar to the implemented system which has been incorporated with Model Sequential CNN. regarding optimizers, only Sanzidul et al. [11] and the executed system have used all three of Adam, ReLU and Softmax. In terms of accuracy on BdSL, Huda et al. [18], Sanzidul et al. [11] and Ishara-bochon [16] have achieved accuracy of 99%, 92.65% and 92.87% respectively whilst the implemented system has achieved 98.45% accuracy. Again in terms of accuracy of ASL, Olamiti et al.[17] and Kohsheen et al.[12] have scored 78% and 98.82% accuracy respectively. Conversely, the executed system has scored 95.05% accuracy. However, only the implemented system has worked on both BdSL and ASL.

### 8.3 Alphabetwise Accuracy

There are 36 alphabets in BdSL. Almost every alphabet has got 100% accuracy but some have not. Here is the list of accuracy of every alphabet (demonstrated in table: 8.3).

Alphabet	Accuracy (%)	Alphabet	Accuracy (%)	Alphabet	Accuracy (%)
অ	94	জ	100	ফ	88
আ	100	ঝ	93	ব	93
খ	88	ট	71	ভ	100
গ	94	ঠ	93	ম	87
এ	93	ড	93	র	100
ও	93	ঢ	100	ল	82
ক	88	ত	100	স	100
খ	94	থ	100	হ	88
গ	88	দ	81	য়	93
ঘ	81	ধ	93	ড়	93
চ	100	ন	100	ং	100
ছ	93	প	100	ঃ	82

Table 8.3: Alphabetwise accuracy of BdSL

For most of the letters (like আ,চ,জ) the implemented system has got the perfect accuracy. Apart from ট, every other letters has scored an accuracy over 80. Among them, most of them are even above 90.

There are 29 signs in ASL. 26 Regular alphabets and 3 extra signs which include del, nothing and space. Many letters don't get the 100% accuracy as the image number in the dataset is very large and it is a global dataset. Here is the list (demonstrated in table: 8.4) of accuracy of every alphabet in ASL.

Alphabet	Accuracy (%)	Alphabet	Accuracy (%)	Alphabet	Accuracy (%)
A	97	K	87	U	47
B	87	L	96	V	92
C	100	M	92	W	86
D	87	N	91	X	97
E	81	O	85	Y	86
F	94	P	100	Z	95
G	99	Q	89	del	96
H	100	R	88	nothing	97
I	100	S	73	space	100
J	100	T	73		

Table 8.4: Alphabetwise accuracy of ASL

The table 8.4 depicts that some letters like H, I, J has got the perfect accuracy. However, Letter U has managed to achieve only 47% of accuracy. Moreover, except S and T which has achieved 73 percentage of accuracy, most of the other letter has accomplished a percentage of eighty plus accuracy. Most of the eighty plus accuracy achieving letters passed 90 plus landmark.

## 8.4 Graph Analysis

In this section, graphs of different prospects have been analyzed in order to demonstrate a more clearer view about the result evaluation using several visual diagrams. The visual diagrams include epoch versus accuracy graph, epoch versus cross entropy loss graph, scatter graph for accuracy and loss, line model graph (accuracy versus loss) and line model for validation (accuracy versus loss).

### 8.4.1 Epoch vs Accuracy graph

For BdSL, In this graph (fig. 8.1) the model accuracy and the Epoch have been shown. The blue line of the graph is the “Train Accuracy” and the red line is the “Validation Accuracy”. The graph displays that when the epoch is very low like 0-1 the train accuracy is very less. After 2 epochs the train accuracy becomes almost consistent and the value is nearly 1 which represents an accuracy near 100%. The validation accuracy is less and it always remains between 0.3-0.6. The peak point of it is at 3 epoch and it rises upto slightly higher than 0.5. After that it again starts to decline and till 4 epoch it has remained over 0.5 and then again started to increase.

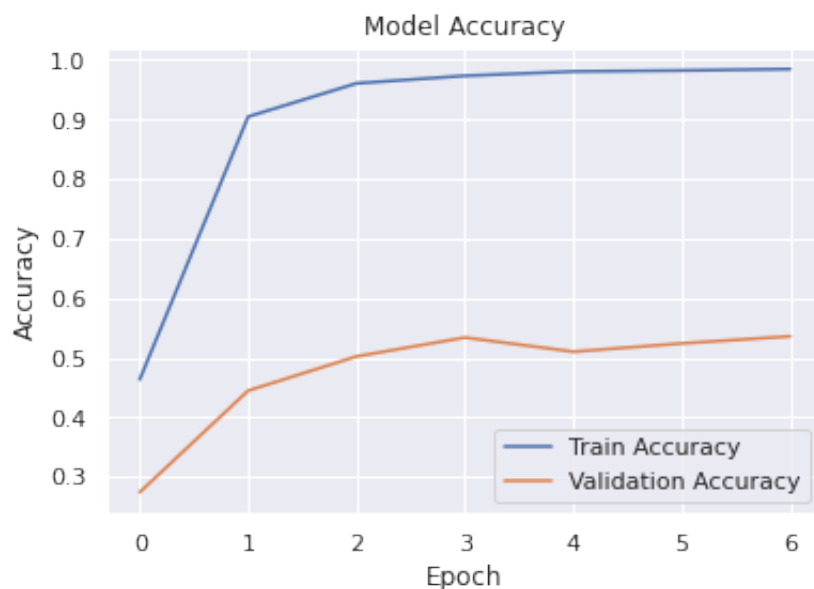


Figure 8.1: Epoch vs Accuracy Graph (BdSL)

For ASL, again in the graph (fig. 8.2) the blue line indicates “Train Accuracy ” and the red line indicates “Validation Accuracy”. The train accuracy in ASL is just like



the train accuracy in BdSL. The value might differ but the value is very near. The validation accuracy is less in case of ASL. It always remains 0.70 to 0.80. In the beginning it is 0.70 and at epoch 4 the validation accuracy is highest and the value is 0.80.

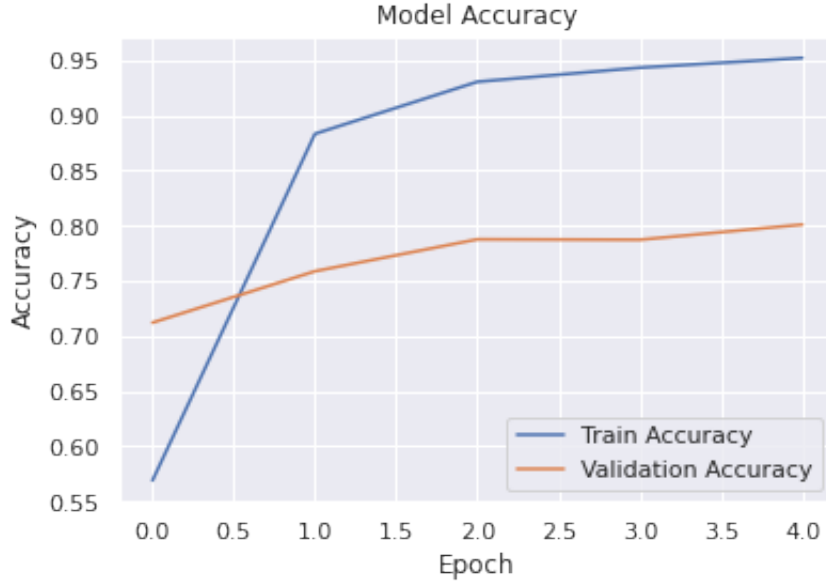


Figure 8.2: Epoch vs Accuracy Graph (ASL)

### 8.4.2 Epoch vs Cross Entropy Loss

Cross Entropy Loss is also known as logarithmic loss. The predicted probabilities for each class are compared to the actual target class [29]. The target score / loss that penalizes the probabilities is calculated based on the distance from the actual expected value. The function for the Cross entropy loss is-

$$L_{CE} = - \sum_{i=1}^n t_i \log(p_i) \quad (8.1)$$

Here,

$n$  = total number of classes

$t_i$  = the label of the truth table consisting of 0 and 1,

$p_i$  = is the soft max probability for  $i^{th}$  classes

For BdSL, This graph (fig.8.3) describes the relation between “Epoch” vs “Cross Entropy Loss”. Here Cross Entropy Loss(y axis) depicts Categorical Cross Entropy Loss which is widely used in multiclass classification. The blue line indicates the “Train Loss” and the red line indicates the “Validation Loss”. In the beginning of the system the train loss is at the highest value. Then it has dropped drastically when the epoch is 1. After that it still fell and from 2 to 6 epoch it is consistent with nearly 0 loss. So the data loss is not there. But there is a chance that this loss can go higher when the epoch value increases. If the system does not overfit the train loss remains close to 0. If it overfits then the train loss will go higher. The

validation loss always gives a value between 2.5-3.5. The lowest validation loss is 2.5 at 2 epoch and the highest validation loss is 3.5 at 4 epoch.

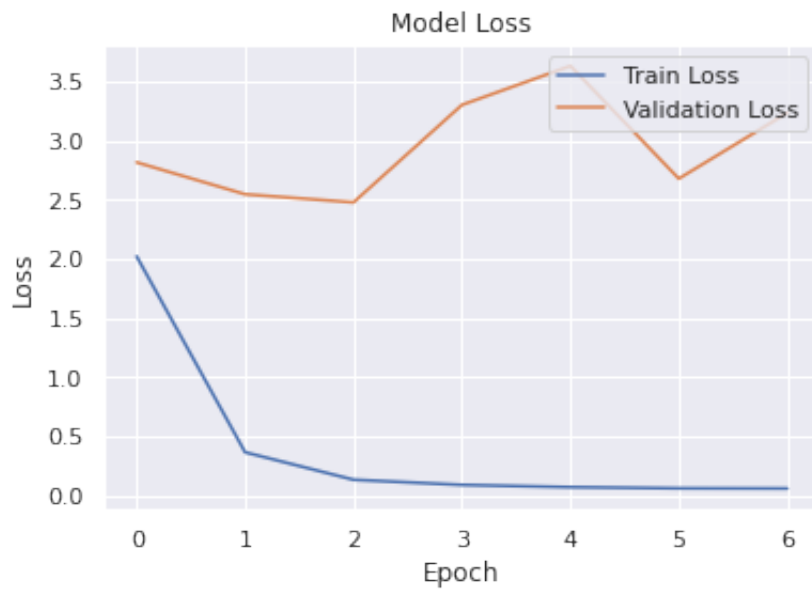


Figure 8.3: BdSL Epoch vs Model Loss

For ASL, The train loss is the same as the train loss of BdSL (fig.8.4). The value may differ but the graph line of the Train Loss is almost similar. The validation loss is higher in this case. The validation loss remains from 1 to 0.8. The lowest it is at 2 epochs. The train loss is lowest at 4 epochs. At 4 epochs the validation loss is slightly higher than 0.8. If the system does not overfit the train loss remains close to 0. If it overfits then the train loss will go higher if the epoch is more just like BdSL.

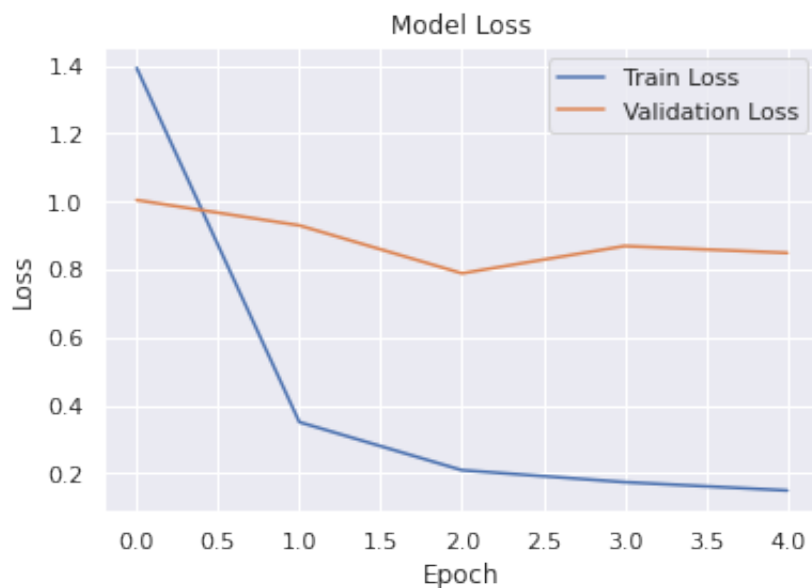


Figure 8.4: ASL Epoch vs Model Loss

### 8.4.3 Scatter Graph for Accuracy and Loss

Two scatter graphs (fig. 8.5) indicate the accuracy and the loss in every epoch. For BdSL, The accuracy in the first epoch is much less than accuracy in any other epoch. So, the loss in epoch 1 is much higher than any other epoch. The highest accuracy is found at the 7 epoch and also the lowest loss is found at 7 epoch.

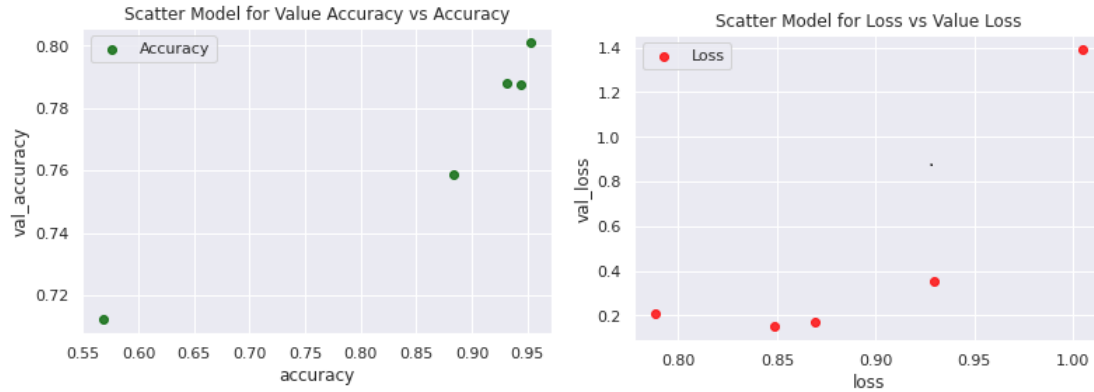


Figure 8.5: Scatter Model BdSL

For ASL, In this part, the system also has lower accuracy at first epoch and the loss is highest at 1 epoch (shown in fig. 8.6). The loss decreases and accuracy increases when the epoch value rises. At epoch 4 the accuracy is highest and the loss is the lowest.

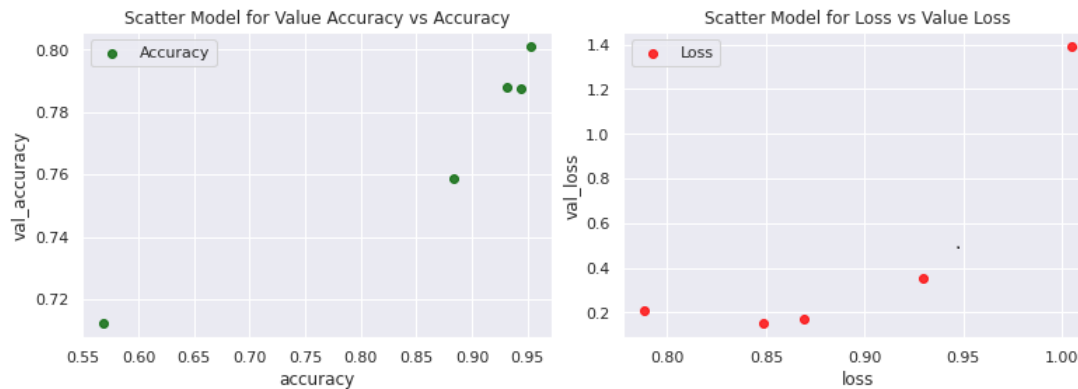


Figure 8.6: Scatter Model ASL

### 8.4.4 Line Model Graph (Accuracy vs Loss)

In this graph (fig.8.7), the x axis is accuracy and the y axis is loss.

For BdSL, The line is a straight line towards down. At first the loss is higher and then accuracy increases. Both are inversely proportional.

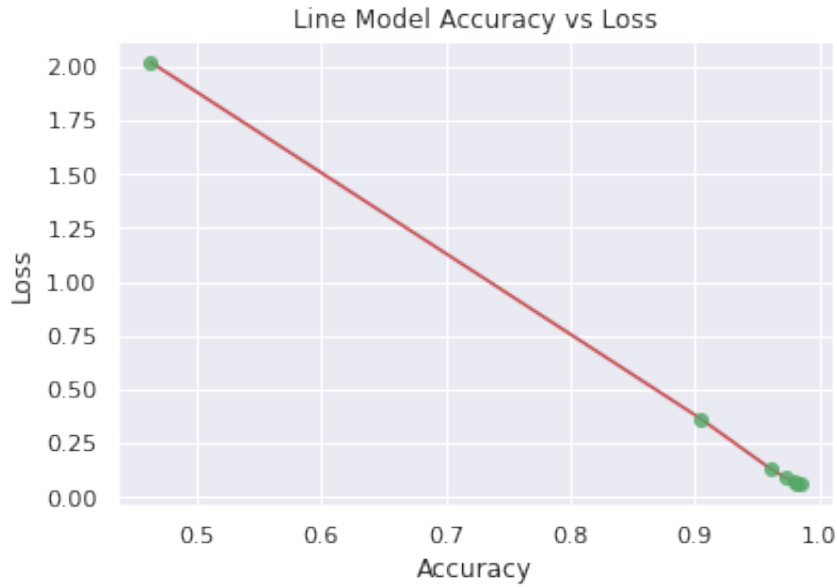


Figure 8.7: Line Model Graph (Accuracy vs Loss) BdSL

For ASL, Again the line is straight towards down (shown in fig.8.8). The loss is higher at first when accuracy increases and the loss decreases. In this case also their relation is inversely proportional.

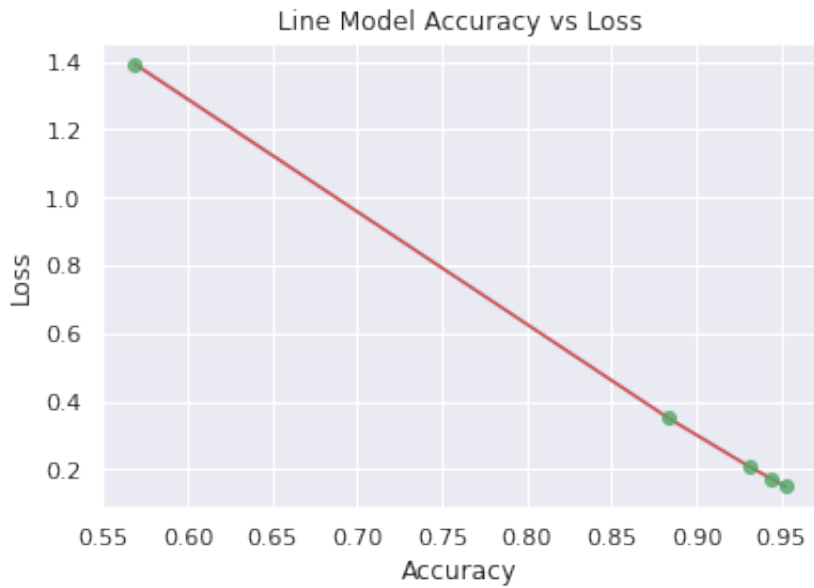


Figure 8.8: Line Model Graph (Accuracy vs Loss) ASL

### 8.4.5 Line Model for Validation (Accuracy vs Loss)

This graph (fig. 8.9) describes the validation points of accuracy vs loss.

For BdSL, At first epoch the validation point is at 2.8 loss and the accuracy is below 0.30. Then at the second epoch accuracy rises and loss decreases. Till the third

epoch accuracy increased and loss decreased. At the fourth epoch there is a drastic change. The accuracy rises and the loss increases to almost 3.2. At 5th epoch the accuracy decreases and the loss increases and again in 6th epoch the loss decreases and accuracy slightly increases. The accuracy increases in the seventh epoch also.

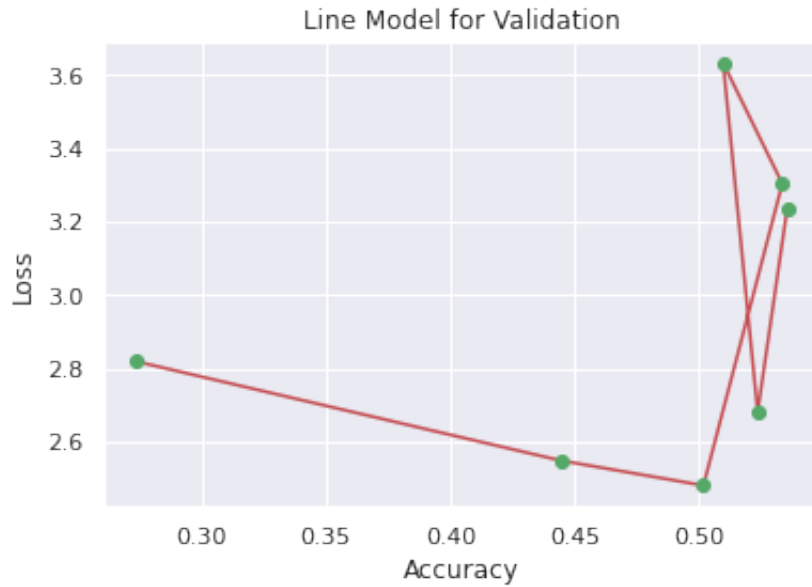


Figure 8.9: Line Model for Validation BdSL

For ASL, Additionally, at first loss is the highest and accuracy is less than 72 (shown in fig. 8.10). Then the loss decreases and the accuracy increases. At epoch 3 the system gives the lowest loss and at epoch 5 the system gives the highest accuracy though here the loss is slightly higher than the epoch 3's loss.

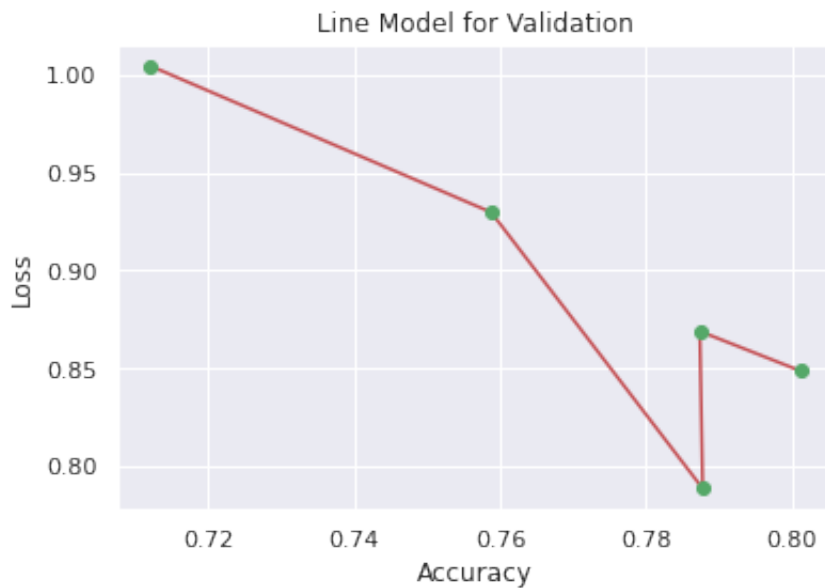


Figure 8.10: Line Model for Validation ASL

## 8.5 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix can also be known as an error matrix because it mainly allows visualization of the performance of an algorithm. The current and incorrect predictions are summarized with count values and broken down by each class.

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} \quad (8.2)$$

$$Precision = \frac{TP}{TP + FP} \quad (8.3)$$

$$Support = \frac{TP}{TP + FN} \quad (8.4)$$

$$F1\_Score = \frac{2 * precision * support}{precision + support} \quad (8.5)$$

Accuracy shows the total number of input divided by total number of output. (shown in. 8.2)

Precision means the ratio of correctly calculated answer vs total predicted answer. (shown in 8.3)

Support means the ratio of true positive [TP] and total positive values. (shown in 8.4)

F1\_score tells the equilibrium steps between precision and support. (shown in. 8.5)

The Main terms of a confusion matrix is false negative [FN], false positive [FP], true positive [TP], true negative [TN].

### ASL Analysis

In this matrix for ASL (shown in fig. 8.11), A is not properly detected. A is slightly mixed up with E (194:78 ratio). I had mistaken precision B,E & K with the ratio of (30 : 82 : 125 : 22). For M, the confusion ratio with other letters are (M : N : S) = (180 : 47 : 16). For N, it is confused with M,S & Z and the confusion rate is (M : N : S : Z) = (56 : 154 : 22 : 17). R is an alphabet which has very low accuracy and is also confused with U. The ratio for R & U is (69 : 37). The confusion ratio of U, V & W is = (60 : 167 : 41). X is mistaken for multiple alphabets like S, T, U and W. But after doing the normalization the confusion matrix is giving very promising results with a little mixed up with other alphabets in some cases specially it fails to detect the difference between R and U. all other ASL are perfectly detected.

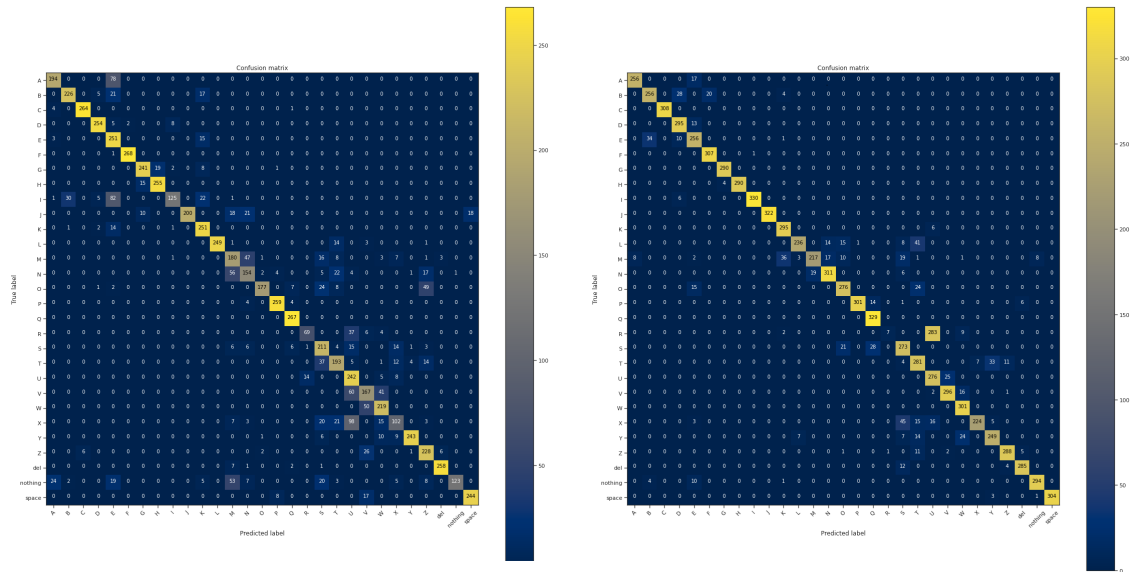


Figure 8.11: ASL Confusion Matrix Before and After Normalization

### BdSL Analysis

For BdSL, Many alphabets showed inconsistent results after training (shown in fig. 8.12). But after doing normalization the accuracy rate increased dramatically and showed almost all the accurate results except ढ. Before Normalization 12 out of 36 alphabets have shown consistent behavior. Also ष and ढ have confusing outputs with different alphabets, ढ, ढ and ढ have many confusing outputs with multiple letters. But after normalization the output is pretty consistent and accurate.

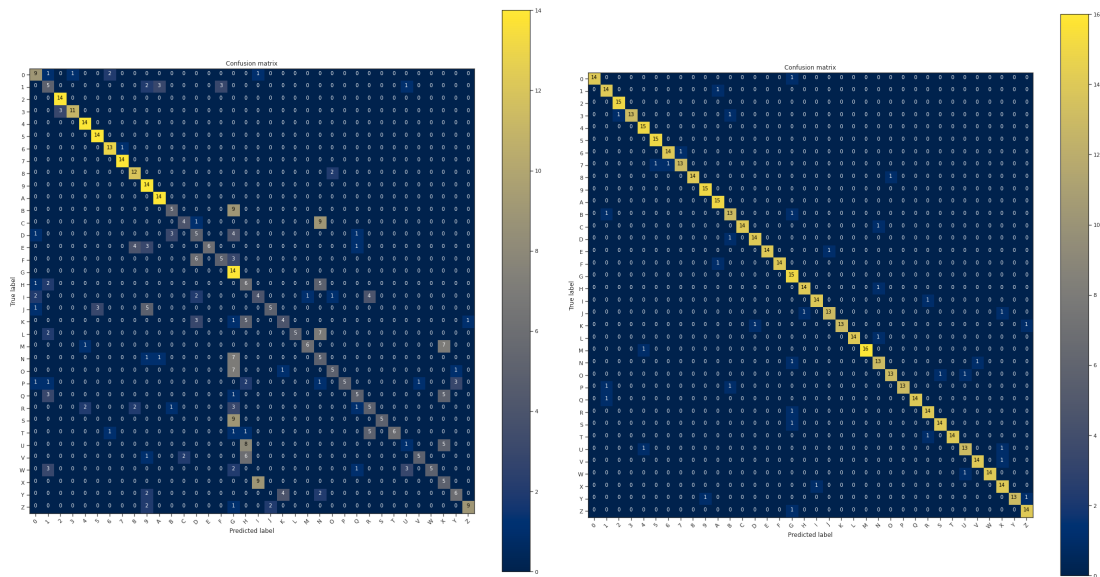


Figure 8.12: BdSL Confusion Matrix Before and After Normalization

# Chapter 9

## Epilogue

### Conclusion

In this paper, a model is built to translate both Bangla and English sign language regarding any kind of background. By manipulating open access datasets for ASL, it has been recognized to us about all the vowels and consonants of the English language. Also a two hand gesture based BdSL dataset has been built using various backgrounds behind the gestures. At the same time, Convolutional Neural Network and Machine learning helped to fit the model. Preprocessing stage paved the way for crop, scale, cross match and optimizers helped to overcome the errors. BdSL is very helpful in Bengali Sign Language but is it hard for recognizing combined and diacritic letters. A developed machine learning algorithm particularly for image processing using CNN is useful for making meaningful sentences. To work in Data Science, increasing the research in developing datasets and more language conversions is necessary. Also the system will develop the communication between the people and will break the barrier. The advantages behind the chosen model is it extracts the image pixels and calculates the best out of Two stride values. Then the flatten layer shapes it for the dense and dense layer makes the best use of the fully connected layers. The optimizers lessen noisy values and show the best output from the train test set. Even with a random background, hand gestures have been extracted successfully and more than desirable accuracy output has been shown. This system can work on 26 English Alphabets on ASL dataset and 36 Bengali Alphabets on BdSL with the same fitting and variable backgrounds with 95.23% accuracy on ASL and 98.45% accuracy on BdSL. The trained model can not only work for a single language, it is trained to work on both Bangla and English hand gestures and recognize it quite accurately. On the contrary, the System can not work with real time input. Also the system can barely differentiate between স and চ, then উ, ব and র these 3 are barely differentiated. ঞ and ণ are also recognized with a 60-40 accuracy. Then ঞ and ঔ have very little accuracy error too. Although the system can detect hand gestures with any backgrounds, it cannot detect hand gestures from less subjective pictures where various elements are present and the hand matrix is very small. However, If the system is applied on a converter it can work accurately and the model is very efficient with any real world backgrounds. It has been planned to work on multiple language conversions other than Bengali and English in the near future. Moreover, it will be very much interesting to work on different algorithms for real time conversions of sign languages such as VGG16, ResNet50 etcetera.



## Future Work Plan

- Since this model has a pretty high accuracy rate even after taking real world input, it has been planned to implement real time conversion using picture extraction from webcam or camera images and cross checking the trained model.
- There has been a discussion to implement VGG16, VGG19, ResNet50, InceptionV3 pretrained models to check if they can show better outputs and make a comparative analysis on that.
- This model is sufficient for detecting two hand gestures with variable background and so primary initiatives have been taken to apply the model for other two hand gesture based sign languages like Hindi or Spanish etc.

# Bibliography

- [1] M. Alauddin and A. H. Joarder, “Deafness in bangladesh,” in *Hearing Impairment*, pp. 64–69, Tokyo: Springer Japan, 2004.
- [2] B. C. Karmokar, K. M. R. Alam, and M. K. Siddiquee, “Article: Bangladeshi sign language recognition employing neural network ensemble,” *International Journal of Computer Applications*, vol. 58, pp. 43–46, November 2012. Full text available.
- [3] B. J. Bahan, “Non-manual realization of agreement in american sign language,” 1996.
- [4] V. Adithya, P. R. Vinod, and U. Gopalakrishnan, “Artificial neural network based method for indian sign language recognition,” in *2013 IEEE Conference on Information Communication Technologies*, pp. 1080–1085, 2013.
- [5] I. McGlenn, “Sign language alphabets from around the world - asl - ai-media,” Sep 2021.
- [6] F. Z. Siddiqua, “The silent conversation,” Mar 2015.
- [7] “American sign language and british sign language differences,” Feb 2022.
- [8] “ANN and CNN: Analyzing differences and similarities.” <https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/>, Feb. 2022. Accessed: 2022-5-15.
- [9] P. Mishra, *Medium*. *Why are Convolutional Neural Networks good for image classification?* 2019.
- [10] M. Rahaman, M. Jasim, M. Ali, and Hasanuzzaman, “A real-time appearance-based bengali alphabet and numeral signs recognition system,” pp. 19–26, 01 2017.
- [11] M. Sanzidul Islam, S. Sultana Sharmin Mousumi, N. A. Jessan, A. Shahariar Azad Rabby, and S. Akhter Hossain, “Ishara-lipi: The first complete multi-purpose open access dataset of isolated characters for bangla sign language,” in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–4, 2018.
- [12] K. Tiku, J. Maloo, A. Ramesh, and I. R., “Real-time conversion of sign language to text and speech,” in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 346–351, 2020.

- [13] R. Shahriar, A. Zaman, T. Ahmed, S. M. Khan, and H. Maruf, "A communication platform between bangla and sign language," in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1–4, 2017.
- [14] S. T. Ahmed and M. A. H. Akhand, "Bangladeshi sign language recognition using fingertip position," in *2016 International Conference on Medical Engineering, Health Informatics and Technology (MediTec)*, pp. 1–5, 2016.
- [15] J. Ravikiran, K. Mahesh, M. Suhas, D. Dheeraj, S. Sudheender, and N. Pujari, "Finger detection for sign language recognition," *Lecture Notes in Engineering and Computer Science*, vol. 2174, 03 2009.
- [16] M. S. Islam, S. Sultana Sharmin, N. Jessan, A. S. A. Rabby, S. Abujar, and S. Hossain, *Ishara-Bochon: The First Multipurpose Open Access Dataset for Bangla Sign Language Isolated Digits*, pp. 420–428. 07 2019.
- [17] V. Adewale and A. Olamiti, "Conversion of sign language to text and speech using machine learning techniques," *JOURNAL OF RESEARCH AND REVIEW IN SCIENCE*, vol. 5, pp. 58–65, 12 2018.
- [18] M. N. Huda and I. Alam, "Automated bangla sign language conversion system: Present and future," 2020.
- [19] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.
- [20] A. Jarman, M. S. Arshad, N. Alam, and M. J. Islam, "An automated bengali sign language recognition system based on fingertip finder algorithm," *International Journal of Electronics and Informatics*, vol. 4, 07 2015.
- [21] K. Deb, M. I. Khan, H. P. Mony, and S. Chowdhury, "Two-handed sign language recognition for bangla character using normalized cross correlation," *Global journal of computer science and technology*, vol. 12, 2012.
- [22] D. Aryanie and Y. Heryadi, "American sign language-based finger-spelling recognition using k-nearest neighbors classifier," in *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, pp. 533–536, 2015.
- [23] M. M. Hasan, M. Khaliluzzaman, S. A. Himel, and R. T. Chowdhury, "Hand sign language recognition for bangla alphabet based on freeman chain code and ann," *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, pp. 749–753, 2017.
- [24] M. T. Hoque, M. Rifat-Ut-Tauwab, M. F. Kabir, F. Sarker, M. N. Huda, and K. Abdullah-Al-Mamun, "Automated bangla sign language translation system: Prospects, limitations and applications," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 856–862, 2016.

- [25] K. K. Podder, M. Chowdhury, Z. Mahbub, and M. Kadir, “Bangla sign language alphabet recognition using transfer learning based convolutional neural network,” *The Bangladesh journal of scientific research*, vol. 31-33, pp. 20–26, 10 2020.
- [26] J. Uddin, F. Arko, N. Tabassum, T. Trisha, and F. Ahmed, “Bangla sign language interpretation using bag of features and support vector machine,” pp. 1–4, 12 2017.
- [27] F. M. J. Mehedi Shamrat, S. Chakraborty, M. M. Billah, M. Kabir, N. S. Shadin, and S. Sanjana, “Bangla numerical sign language recognition using convolutional neural networks (CNNs),” *Indones. j. electr. eng. comput. sci.*, vol. 23, p. 405, July 2021.
- [28] Prabhu, “Understanding of convolutional neural network (CNN) — deep learning.” <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>, Mar. 2018. Accessed: 2022-5-21.
- [29] K. E. Koeh, *Towards Data Science. Cross-Entropy Loss Function*. 2020.

# Appendix

## **Real Time Gesture Detection (for Future Implementation)**

From the very beginning of the research plan, implementing proposed model in real time was the goal. To materialize this process, images were extracted from a running video through a webcam or any working camera. The following working process is described below.

### **Real Time Input Collection**

#### **Turn the camera on**

Taking input is the first step of this simulation. When the user will start the program input will be taken from a real time image using the camera. Therefore, the camera will be turned on. Here enough light is mandatory and the user must show the appropriate gesture through the camera.

#### **Requesting users to show gestures**

This system has been designed to be simple and user friendly so that anyone can use and understand what is actually going on. When the camera is on, the user will be shown a request message by the program to give the input in the remarked frame.

#### **Capturing gesture**

Universal resolution images are taken as input to perform the task fast and to match the input with the train and test set picture resolution. But all users will not be able to give an input of exact quality that is wanted. So an image frame is designed which will take the required resolution image. User just has to give his input into that frame and then it will be auto captured by the program. For each change of gesture, the program will take different inputs and work accordingly. If the user gives a termination signal, the program will be terminated and stop taking inputs.

#### **Passing acknowledgement to user**

In the next step, the program sends an input status message to the user for acknowledgement. It is quite natural that a user can give a blurry image or can capture an image in low light. In that case the input might not be suitable for the further processes. So the program is proposed in such a way that it will give the user the status of the input. For any invalid, shaky or blurry image the program is supposed to show a warning message. However, if the user gives an appropriate input then the program will show an input status for the acknowledgement.

#### **Pre-processing of Real Time Input**

After taking the input, images need to be pre-processed in order to make it suitable for cross checking with the dataset and model. In order to do that preprocessing will be done exactly the same way like the dataset preprocessing time. After the preprocess input will be sent for cross-checking with the proposed model.

#### **Cross-Checking Preprocessed Input with Model**

When the preprocessing of an input will be finished, the system will cross check the preprocessed image with the model.

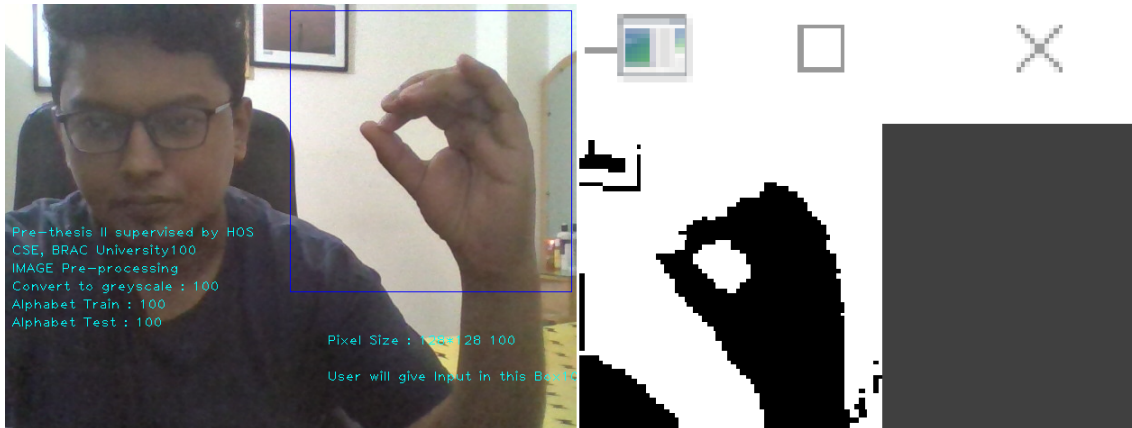


Figure 9.1: Gesture of Alphabet “O” in the input area & taking Input in grayscale form

### Prediction

Although symbol 'O' has been detected properly, for most of the other symbols, expected accuracy result did not come up. So the accuracy was below 15% in real time.

### Accuracy Assumption

As for the moment it has not been possible to get a good result in terms of accuracy in real time, it will be an idea for the further future works.

### Issues with real time data

While working with real time data we have examined and analyzed the following facts are examined and analyzed:

#### Unnecessary image in the input frame

In the figure 9.1 it can be seen that there is a picture frame in the background. This is an unnecessary part for the input but the program has also preprocessed it (shown in fig. 9.2). Consequently, it is giving an invalid outcome. It is dropping the accuracy badly. So the unnecessary part from the input has to be reduced.



Figure 9.2: Some noises which should have been reduced

#### Losing image information from real time input

Here in fig. 9.2, it can be seen that some parts of the hand are lost. Though it worked for the symbol C but for complex symbols like, X, K for ASL and ক, খ, চ for BdSL the result was not satisfactory. Sometimes The gesture C and O had a

conflict. So this is why the accuracy decreased in real time. (shown at fig 9.3)



Figure 9.3: Gesture of alphabet “C” and “O” which look quite similar in grayscale form