

# A Novel Lightweight CNN Approach for Bangladeshi Sign Language Gesture Recognition

by

Aryan Rahman

18201174

Ahbab Ali Khan

18201190

Tazwar Mohammed Shoumik

18201121

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

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 material previously published or written by a third party, except where this 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 a university or other institution.
4. We have acknowledged all main sources of help.

## Student's Full Name & Signature:



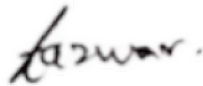
---

Aryan Rahman  
18201174



---

Ahbab Ali Khan  
18201190



---

Tazwar Mohammed Shoumik  
18201121

# Approval

The thesis/project titled “A Novel Lightweight CNN Approach for Bangladeshi Sign Language Gesture Recognition” submitted by

1. Aryan Rahman (18201174)
2. Ahbab Ali Khan (18201190)
3. Tazwar Mohammed Shoumik (18201121)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 24, 2022.

## Examining Committee:

Supervisor:  
(Member)



---

Ms. Arnisha Khondaker  
Lecturer  
Department of Computer Science and Engineering  
Brac University

Program Coordinator:  
(Member)

---

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

Head of Department:  
(Chair)

---

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

## Abstract

The impairment of speech impediment affects 6.9% of Bangladesh's population. This is a condition in which people cannot communicate vocally with others or hear what they are saying, causing them to rely on nonverbal means of communication. For such persons, sign language is a common way of communication in which they communicate with others by making various hand gestures and motions. The biggest problem is that not everyone understands sign language. Many people cannot converse using sign language, making communication between them problematic. Even though translators and interpreters are available to assist with communication, a more straightforward method is required. We propose a method which uses deep learning combined with some computer vision techniques to detect and classify Bangla sign languages to close this gap. Our custom-made CNN model can recognize and classify Bangla sign language characters from the Ishara-Lipi dataset with a testing accuracy of 99.21%. To recognize the precise indications of a hand gesture and understand what they mean, we trained our model with sufficient samples by augmenting and preprocessing the Ishara-Lipi dataset using various data augmentation techniques.

**Keywords:** Bangladeshi Sign Language(BdSL); Deep Learning; Convolutional Neural Network; Image Processing; Image Classification

## **Dedication**

All praise to the Almighty Allah for keeping us safe during the global pandemic. We dedicate our thesis to the millions of people who have trouble communicating with other people because they lack the ability to hear or speak. May their lives be made a bit more better with our work.

## **Acknowledgement**

Firstly, all praise to the Almighty Allah for whom our thesis has been completed without any major interruption.

Secondly, to our beloved and respected supervisor Ms. Arnisha Khondaker for her kind, unwavering support and advice throughout our work.

And finally, to our friends and families for their prayers and well wishes. With their generous support, we are now on the verge of our graduation.

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Dedication</b>	<b>iv</b>
<b>Acknowledgment</b>	<b>v</b>
<b>Table of Contents</b>	<b>vi</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Nomenclature</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Goals . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Research Objective . . . . .	3
<b>2 Literature Review</b>	<b>4</b>
2.1 Core Technologies . . . . .	4
2.1.1 Computer Vision . . . . .	4
2.1.2 Deep Learning . . . . .	4
2.1.3 ANN . . . . .	5
2.1.4 CNN . . . . .	6
2.1.5 Core Libraries . . . . .	8
2.2 Bangladeshi Sign Language Datasets . . . . .	9
2.2.1 Single handed sign language datasets . . . . .	9
2.2.2 Double handed sign language datasets . . . . .	10
2.2.3 Static Datasets . . . . .	10
2.2.4 Dynamic Datasets . . . . .	10
2.2.5 Open Source Datasets . . . . .	10
2.3 Related Works . . . . .	11
2.3.1 Fuzzy Logic . . . . .	12
2.3.2 Artificial Neural Network . . . . .	12
2.3.3 Principal Component Analysis . . . . .	13

2.3.4	Hidden Markov Model . . . . .	13
2.3.5	K-Nearest Neighbor . . . . .	13
2.3.6	Support Vector Machine . . . . .	14
2.3.7	Convolutional Neural Network . . . . .	14
2.3.8	Contour Analysis . . . . .	14
<b>3</b>	<b>Methodology and System Implementation</b>	<b>16</b>
3.1	Dataset . . . . .	16
3.1.1	Dataset Collection . . . . .	16
3.1.2	Data Preprocessing . . . . .	18
3.2	Model Overview . . . . .	21
3.2.1	Intuition Behind the Model . . . . .	21
3.2.2	Model Architecture . . . . .	21
3.2.3	Model Efficiency Considerations . . . . .	22
3.3	System Workflow . . . . .	23
3.4	Model Training and Testing . . . . .	25
3.4.1	Overfitting & Under fitting . . . . .	25
3.5	Model Deployment Considerations . . . . .	25
<b>4</b>	<b>Experimental Results and Analysis</b>	<b>26</b>
4.1	Experimental Results and Analysis . . . . .	26
4.2	Model Accuracy and Loss . . . . .	26
4.3	Model Evaluation . . . . .	27
4.3.1	Confusion Matrix . . . . .	27
4.3.2	Other Metrics . . . . .	28
<b>5</b>	<b>Conclusion</b>	<b>30</b>
5.1	Conclusion . . . . .	30
5.2	Practical Implications . . . . .	30
5.3	Future Work . . . . .	30
	<b>Bibliography</b>	<b>35</b>



# List of Figures

2.1	Architecture of an Artificial Neural Network . . . . .	5
2.2	Mathematical Computations inside a Neuron . . . . .	6
2.3	Visual Representation of a CNN Model . . . . .	6
2.4	An image being convolved in the convolution layer . . . . .	7
2.5	Visual Representation of Max Pooling . . . . .	7
2.6	Pie chart of research work based on different datasets in BdSL . . . . .	9
3.1	Original Samples from IsharaLipi Dataset . . . . .	16
3.2	One sample from each class of the original dataset . . . . .	17
3.3	Data Augmentation Workflow to produce 1000 samples of images per class where P is the probability of an augment operation . . . . .	18
3.4	Class distribution of augmented samples . . . . .	19
3.5	One sample from each class of the augmented dataset . . . . .	20
3.6	Summarized Plot and Propagation from Layer to Layer . . . . .	22
3.7	Workflow of creating, tuning and building our custom model . . . . .	24
4.1	Transition of Model Accuracy over 10 epochs . . . . .	26
4.2	Transition of Model Loss over 10 epochs using Categorical Crossentropy	27
4.3	Confusion Matrix for our Custom Model . . . . .	28

# List of Tables

3.1	Sequential architecture and hyperparameters of our proposed model .	22
3.2	Parameter count of our proposed model . . . . .	23
4.1	Classification Report on Testing Dataset . . . . .	29

# Nomenclature

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

ANN Artificial Neural Network

ASL American Sign Language

AUSLAN Australian Sign Language

BdSL Bangladeshi Sign Language

BSL British Sign Language

CDD Centre for Disability in Development

CNN Convolutional Neural Network

D&D Deaf and Dumb

GS Austrian Sign Language

HMM Hidden Markov Model

HSC Hand Sign Classification

JSL Japanese Sign Language

KNN K-Nearest Neighbor

LMC Leap Motion Controller

LSF French Sign Language

NNE Neural Network Ensemble

PCA Principal Component Analysis

ROI Region Of Interest

SVM Support Vector Machine

# Chapter 1

## Introduction

For a long time, computer vision has been a field of research. However, with the introduction of deep neural networks, this progress has accelerated. It is a process in which a machine is trained to recognize specific aspects from video or images provided to it as data and then performs various tasks based on the aspects detected in the data. This may be found in almost everything, from the simplest to the most sophisticated. It's even included in the face detection that we use on our phones, where the phone's camera is used to save different elements of our faces, and then our faces are matched with the saved image every time we unlock the phone using facial recognition. Again, we see the use of computer vision in complex things like self-driving cars, where the machine is fed data and taught in such a way that it can make instantaneous choices, ensuring that no accidents occur.

To detect and infer the identified signs into the desired language, most existing systems use either a vision-based or a sensor-based technique. Using a video source, vision-based techniques gather images or video of hand gestures. Sensor-based techniques, on the other hand, use sensors and instruments like gyroscopes, accelerometers, flex sensors, Microsoft Kinect, ultrasonic, mechanical, and others to determine the location, motion, and velocity of the hands. In recent years, research on recognizing BdSL has made great progress and shown promising results.

### 1.1 Motivation and Goals

Nowadays, computer vision is not only limited to one or two sectors and is being used widely, even for small pet projects, since information regarding this is publicly and widely available. One of the problems that computer vision is also used for is sign language detection. As hearing and speech-challenged people are unable to communicate orally or through other ways, sign language is the most important means of communication for them. A considerable percentage of people suffer from speech impediments; they rely on sign language for communication. Hand gestures, bodily postures, and face emotions of various kinds are used in this communication approach, each communicating a different meaning. American Sign Language (ASL), Japanese Sign Language (JSL), British Sign Language (BSL), Austrian Sign Language (GS), Bangladeshi Sign Language (BdSL), Australian Sign Language (AUSLAN) and other sign languages are used all over the world. There are several sorts of sign language representations: single-handed, double-handed, static, and dynamic.

1 Sign language solves the problem to a certain extent, but another problem that persists is the fact that not a lot of people understand sign language and because of that, there needs to be a way so that both parties understand what the other is saying for swift communication. An intermediary who can translate for both is a solution for this and that is exactly what we are proposing for this paper, but the purpose of our work is that instead of the intermediary being a person, we will automate it using computer vision where a computer can understand and classify sign language to Bengali alphabets.

## 1.2 Problem Statement

Sign language is the mode of communication for the deaf and dumb(D&D). A universal sign language does not exist and just like spoken languages, differs from region to region. Sign languages are also separate languages, consisting of its own grammar and linguistic rules. Approximately 70 million deaf and general people in the world use sign language as their primary mode of expression. In particular, Bangladesh has around 2.6 million individuals who are not able to communicate by speaking [28]. General people who are fluent in sign language can translate it for those who do not know sign language, but it would be a very tedious process. The D&D people cannot even understand the lip reading of the general people, which means there is no alternative to sign language for the D&D to communicate with regular people [1]. Furthermore, for essential communication, D&D people do not usually prefer to write the standard text because the composition of the sign language differs from that of the normally written text [4]. As a result, the D&D people are often ignored in many situations, creating a severe communication gap in society. To eradicate this gap, language and speech learning institutions in Bangladesh mostly follow the formal sign language developed by the Centre for Disability in Development (CDD). This will allow better communication to take place between the D&D and the general people as the formal sign language ensures a centralized communication platform for both parties.

The field of machine learning combined with computer vision is advancing and reaching into various sectors to help ease human life. Nowadays, it is being used to help the D&D community by improving the sign language recognition techniques. There are multiple sign languages all over the world of which some of them are American Sign Language (ASL), British Sign Language (BSL), French Sign Language (LSF) and Japanese Sign Language (JSL), all of which were independently architected and are different from each other [27]. Hence, Bangladeshi Sign Language (BdSL) also differs from others, so it will also require a different dataset for a sign language detector to train the model of BdSL. This requirement of separate datasets for different sign languages has created scarcity in datasets in this area of research. Hence, some researchers have come up with their own customized datasets which cater to their specific objectives.

## 1.3 Research Objective

Here we discuss the primary objectives that we hope to achieve through our research. First of all, we obtain multiple open-access datasets for our problem across and choose the dataset most suitable to use. Then we apply various pre-processing and augmentation techniques to the images in the dataset. Afterwards we train, validate and test our CNN model on the chosen dataset with the pre-processed and augmented images. This will ensure that our model performs in various settings and is versatile enough to be used under different spatial scenarios and computational environments. We aim to achieve a reasonable accuracy for our model over our chosen dataset. A higher accuracy would be preferred but not essential as we are exploring new, efficient techniques and additions to our CNN model.

Lastly, we aim to build a model that can provide results better than the past iterations of BdSL detection models. This means our model would need to require less computational effort, potentially translating to lower latency in detecting hand gestures. Combined with our model, these open-access datasets will enable future researchers to work on our problem and provide further iterations to our approach to solving this critical problem. This dataset will also be used to train, verify and test our new model to ensure that it reaches the same accuracy levels as the other datasets.

# Chapter 2

## Literature Review

### 2.1 Core Technologies

This section will present some of the fundamental technologies and concepts used in our research methodologies. Among the mentioned technologies and concepts, there are also many others used that we do not feel the necessity to explicitly mention and define as they have been in use in this area of research for an extended period and are not “core” to this particular experiment. Some examples of such which are not mentioned but have been used for this research are Artificial Intelligence, Machine Learning, Matplotlib, NumPy and pandas.

#### 2.1.1 Computer Vision

Computer vision refers to any methodology or technology using which computers or intelligent machines gather important information from images, videos or other visual inputs. Using this information, the computer can either choose to gather it for further analysis or can be used to take actions based on those inputs. In order to train a computer vision machine, a lot of visual data has to be fed to it, after which it can decide on its own what it needs to do. The decisions these machines have to make are spontaneous, for which the data it is taking has to be analyzed using fast algorithms and this can only be achieved with proper training of the machine. Image classification, object detection and image segmentation are typical examples of a computer vision problem that can be solved by machine learning algorithms such as Deep Learning, SVM, etc.

#### 2.1.2 Deep Learning

Deep Learning is a subset of Machine Learning which uses different algorithms inspired by the human brain or, in other words, neural networks. That means it imitates how human beings gain knowledge and uses it to teach a machine. Using these algorithms, the machine learns from a large amount of data and this data helps the machine to gradually improve the outcome that it gives. Deep learning is being used in different sectors nowadays, for example, customer experience, Text generation and Medical Research. Essentially a very deep neural network is referred to as Deep Learning.

### 2.1.3 ANN

Biological neural networks establish the structure of the human brain, and the phrase “Artificial Neural Network” is taken from them. Artificial neural networks, like the human brain, include neurons that are coupled to one another at various levels of the networks. From a mathematical standpoint, a neural network is a machine learning algorithm that uses a network of composite mathematical functions to understand and translate input data into some form of output. Figure 2.1 shows a visual representation of a very simple neural network with three types of layers: input layer, hidden layer and output layer. This essentially means that neural networks are nothing but a massive and complex composite function with the ability to approximate almost any form of function, which can produce very accurate results for many use-cases.

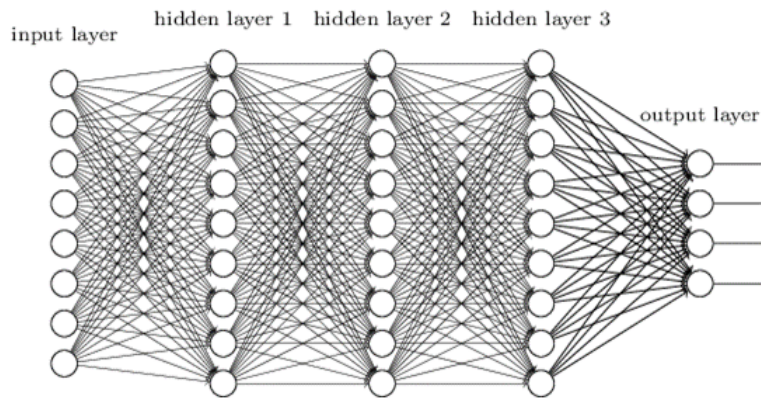


Figure 2.1: Architecture of an Artificial Neural Network

#### Input Layer

Input nodes are the inputs from the outside world that the model uses to learn and draw conclusions. The information from input nodes is passed to the next layer, which is the Hidden layer. It takes data in a variety of formats specified by the programmer.

#### Hidden Layer

The input data from the previous layer is processed by the hidden layer, consisting of a group of neurons. One or more hidden layers can exist in a neural network. The most basic neural network comprises only one hidden layer. If we have a closer look inside a neuron in figure 2.2, we can see many mathematical notations and calculations being performed. This particular neuron has three inputs and weights and their weighted sum is being calculated using equation 2.1. This weighted sum is then passed through an activation function to produce the output of this neuron. This process is repeated for every neuron in the hidden layer(s) in a neural network.

$$\sum_{i=1}^n W_i * X_i + b \quad (2.1)$$



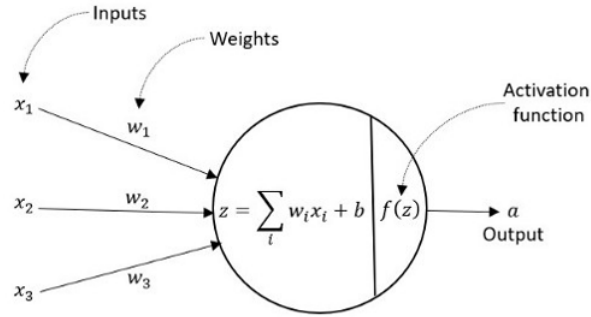


Figure 2.2: Mathematical Computations inside a Neuron

## Output Layer

The output layer is the model's conclusion, which is generated from all of the preceding layers' computations. The output layer might have a single or several nodes. The number of output nodes in a binary classification problem is only one; however, the output nodes might be more than one in a multi-class classification problem.

### 2.1.4 CNN

Convolutional Neural Network is a type of multilayer perceptron network commonly used in computer vision and, in certain situations, natural language processing. The layers of a CNN are as follows: an input layer, output layer, and hidden layer(s) with numerous convolutional layers, pooling layers, fully connected layers, and optional normalization or batch normalization layers. When an image is fed as input to CNN, each layer of the model generates many activation maps, also known as feature maps. These feature maps emphasize the most important aspects of an image. Basic characteristics such as horizontal, vertical, and diagonal edges are usually detected by the initial layers of the CNN(layers close to the input layer). The output of the preceding layer is fed to the next layer as input, which extracts much more complicated features like corners and edge combinations. The deeper a convolution layer is in a convolutional neural network(closer to the output layer), the more complex features it detects like objects, faces, and other things. Figure 2.3 shows a typical representation of a CNN model. CNNs are much more computationally efficient and accurate than ANNs in image classification due to their ability to only learn important image features that matter when classifying between different images.

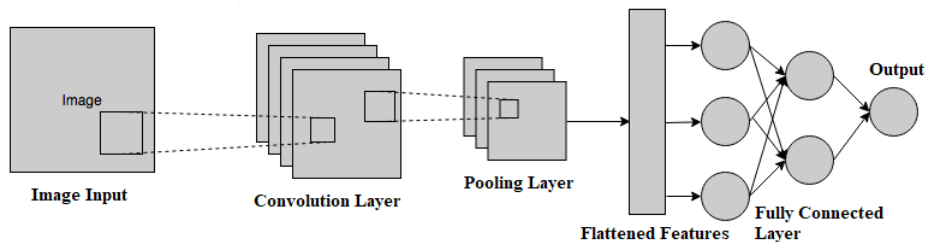


Figure 2.3: Visual Representation of a CNN Model

## Convolution Layer

The convolutional layer is the backbone of a CNN and it is where the layer works to build filters to decipher certain features from images. It needs input data and a kernel filter, which produces feature maps of an image containing the features used to recognize objects of interest. The kernel filter slides through the whole array of pixels within the image to extract meaningful features such as edges and curves. This methodology is termed as convolution. This is done by calculating the dot product of the input pixels and the kernel values according to figure 2.4. The final output is a set of feature maps containing useful features detected from the image.

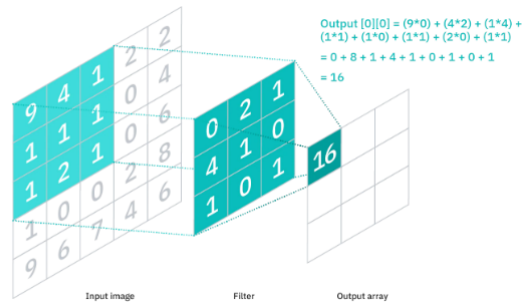


Figure 2.4: An image being convolved in the convolution layer

## Pooling Layer

Pooling layers downsample or reduce the dimensionality of the input from the convolution layer. The pooling process slides a kernel filter across the entire feature map, similar to the convolution layer. However, the maximum or average pixel value is taken from the kernel filter instead of taking the dot product to reduce the input's dimensions. There are two types of pooling: max pooling and average pooling. They aid in reducing complexity, increasing efficiency, and reducing the possibility of overfitting. Figure 2.5 shows how a 2x2 filter of stride 2 downsamples a 4x4 input image into 2x2.

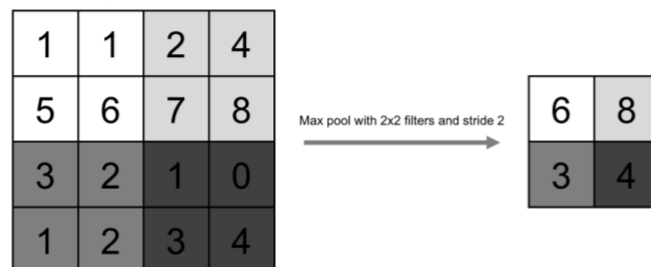


Figure 2.5: Visual Representation of Max Pooling

## Fully Connected Layer

This layer performs classification tasks based on the feature maps, which are fed in as input by preceding convolutional layers. This layer is essentially an ANN that

acts as a classifier. After the pooling layer, the pixels are flattened to a 1-D vector in figure 2.3 which is analogous to ANN. This flattening layer acts as an input layer for the classifier, where it contains the features extracted from the convolution and pooling layers in the previous stages of the model. This flattened layer is usually connected to one or more hidden layers with an output layer at the end.

### 2.1.5 Core Libraries

This section describes some of the fundamental libraries that we employed in our research. Some of the data visualization libraries we used, which are not “core” to this research, such as Matplotlib, seaborn and OpenCV are not explicitly mentioned in the following subsections.

#### **Augmentor**

Augmentor is a Python library that helps with image data augmentation and artificial generation for machine learning applications. It primarily serves as a data augmentation tool and provides elementary image pre-processing capabilities. To augment images, the Augmentor library employs a stochastic, pipeline-based technique. This pipeline technique allows users to connect augmentation operations like skew, shears, rotations, zooming and cropping together and feed images through the pipeline to generate new augmented data. All operations in the pipeline are stochastically applied, both in terms of the likelihood of the operations being applied to each image as it goes through the pipeline and in terms of the parameters of each operation, which are also randomized within user-defined ranges [14]. This allows sampling from a set of potential images created at runtime by the pipeline. The Augmentor library is available to use on Python and Julia only. Source codes are available on GitHub and complete documentation is available on their official webpage.

#### **TensorFlow**

TensorFlow is an open-source machine learning library. Machine learning and deep learning applications are implemented with it. The Google team designed TensorFlow to explore and investigate exciting artificial intelligence concepts. Since TensorFlow is written in Python, it is a very user-friendly library for small pet projects. It uses Python to create a front-end API for developing applications, which is then executed in the super-fast C++ language.

#### **Keras**

Keras is a high-level Python API for building neural networks. It is based on open-source machine learning libraries such as TensorFlow, Theano, and the Cognitive Toolkit. Since it is built on Python, it gives a high-level Python frontend flexibility of different backends for computation, making its implementation user-friendly. However, this makes Keras slower than other deep learning frameworks. Keras has been chosen as TensorFlow’s official high-level API and is now embedded on TensorFlow.

## 2.2 Bangladeshi Sign Language Datasets

One of the most essential steps in ensuring the model’s versatility and robustness is selecting the dataset used to train it. The model’s ability to effectively train itself and attain high validation and test accuracies is dependent on the size and diversity of the dataset. However, gathering this information is a time-consuming operation requiring a significant amount of human labor. With numerous data augmentation tools, this work has become more accessible. In the context of computer vision data, data augmentation entails skewing, rotating, translating, and a variety of other transformations to add variation to the data. We may also alter the background and add image artifacts to help the model recognize photos that have been warped in some way.

Before taking a look at the different datasets available for Bangla Sign Language, let us first look at how they can be categorized. In general, sign language datasets can be categorized into four categories: single-handed, double-handed, static, and dynamic. Single-handed and double-handed are sign language categories based on the hands used in the communication process. At the same time, static and dynamic are categories based on the type of input to the machine. Figure 2.6 shows a visual representation of the distribution of current datasets of BdSL in terms of single-handed and double-handed.

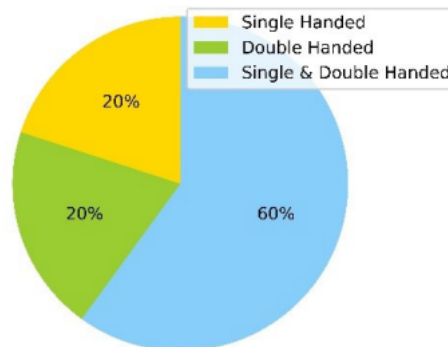


Figure 2.6: Pie chart of research work based on different datasets in BdSL

### 2.2.1 Single handed sign language datasets

Single-handed sign language is a type of sign language where in order to communicate, only one hand is used. For this sort of communication, people use their dominant hand mainly. This is not as common but there are datasets as well as people who communicate in sign language using only their hand. When Control-Engine and Microsoft Voice Command take user input and turn it into text, this is one example. The text is cross-referenced with existing annotations, and a three-dimensional graphical hand motion is displayed in accordance with the text. However, the intelligent assistant’s precision is 82 percent, with the system understanding just discrete phrases and displaying only one hand signal.[23]

## 2.2.2 Double handed sign language datasets

This is a more common mode of communication among the people who use sign language for communicating. Here, both hands are used to show different signs and facial expressions used for communicating. This type of dataset is utilized in systems that assess the angles between different hand and body portions and compare them to values in the database. These values are saved while manually teaching the system with various hand movements.[23]

## 2.2.3 Static Datasets

These include static images with no time frame. Static datasets have images of finger spelled signs and these types of datasets can be either single or double-handed. These datasets have specific machines that can interpret them and be used early on in the Bangla sign Language recognition process. Based on the findings, there are seven systems that can recognize static BdSL gesture graphics. Four can detect static two-handed hand movements, whereas two others can distinguish static single-handed hand gestures.

## 2.2.4 Dynamic Datasets

These are continuous and real-time hand motion inputs. Different letter instances cannot be compared using typical Euclidean space when comparing due to temporal deviation. We will need to employ advanced filtering and algorithms to compute similarity likelihood. These kinds of datasets were made in order to get over the shortcoming of the static datasets and the inaccuracy and inefficiency created due to the static datasets.

## 2.2.5 Open Source Datasets

Moving on to the available datasets of BdSL that are open-source, we can see a few that we can work with. The datasets available can be categorized into single-handed and double-handed and sometimes a mixture of both. 60% of the datasets are a mixture of both single-handed and double-handed, while the rest 40% is divided equally at 20% each for single-handed and double-handed, as we see from the figure. Mentioned below are the different open source datasets that are available for BdSL.

### BdSL-D1500

In this dataset, Bengali sign alphabets, numbers, and gestures are represented in both one-hand and two-hand versions. There are a total of 87 classes. There are 38 classes for one-hand BdSL representation, 36 classes for two-hand BdSL representation, 12 classes for numerals, and one class for a miscellaneous gesture. There are around 1517 RGB photos in each class [26].

### BdSL36

To realize the real-life use case of a BdSL recognition, the authors of BdSL36 set out to create a versatile dataset with over 4 million images that mimic a person's situation using the model with their device. BdSL was created with the help of 10

volunteers, with 1200 initial images however, with the use of data augmentation, the number of images to be used increased to 4 million. BdSL36 stands out from the other datasets as it is built outside of a controlled lab environment [21].

### **IsharaLipi**

The authors of Ishara Lipi saw a lack of available open access datasets for the sole purpose of BdSL recognition. Most papers that worked on this problem used their own dataset that was created by them and did not make it publicly available for use in research. IsharaLipi [17] incorporates scripts to manipulate the images by cropping them, resizing them and converting them to grayscale. This dataset contains 50 sets for each of the 36 signs found in BdSL.

### **KU-BdSL**

The KU-BdSL(Khulna University Bengali Sign Language dataset) [22] is a Bengali sign language dataset with three versions. The versions are as follows: Uni-scale Sign Language Dataset (USLD), Multi-scale Sign Language Dataset (MSLD) and Annotated Multi-scale Sign Language Dataset (AMSLD). Images representing single-hand motions for BdSL alphabets make up the dataset. Many different smartphones were used to acquire photographs from 33 individuals(25 males and 8 females). Each version has 30 classes that correspond to the 39 Bengali consonants. Each version of the dataset has a total of 1,500 samples in jpg format. To have different brightness and contrast, the images were clicked on uniform surfaces at varying times of the day. Each picture in USLD is 512\*512 pixels in size. The majority of examples in this dataset had the intended hand position in the middle. In MSLD, the raw photos are saved so that researchers can edit the dataset according to their needs. AMSLD has multi-scale annotated data, which could be helpful in tasks such as classification and localization. YOLO DarkNet annotation was used to annotate the images in the dataset.

## **2.3 Related Works**

This section will look into some previous and existing research on this topic. Initially, we will discuss the origins of Sign Language Recognition and notable breakthroughs in the field. Then we will move on to more novel approaches that are state-of-the-art and bring significant impact on the world, which are divided into subsections according to different technologies used. The previous technologies that we will see have used both the same and different approaches compared to our approach.

Existing researches of sign language detection techniques use various machine learning algorithms such as fuzzy logic [2], Artificial Neural Network [12] [8], Principal Component Analysis (PCA) [10] [5], Hidden Markov Model (HMM) [6] [3], K-Nearest Neighbor (KNN) [9], SVM [13], Convolutional Neural Networks (CNN) [15] [18] [19] [25] [20] and Contour Analysis [11]. to train the model using existing or customized datasets. All the mentioned algorithms have their benefits and drawbacks in terms of their time and space complexity. This implies that some algorithms may perform well in certain situations while others may perform poorly in others. As

a result, it all comes down to the situation at hand, namely the picture quality of the sign language and the datasets used to train the model to recognise certain sign languages.

Much work has been done on the use of CNNs and gesture recognition as a result. The image-based approach used in one such paper created their own dataset of images containing the Bengali sign gestures and implemented a VGG19 based CNN which eventually achieved a test accuracy of 89.6%. A similar challenge was faced in this paper as the researchers were working on a limited number of datasets, while CNNs typically can give better results when they train on very large datasets. Another paper introduced a real-time computer vision-based Bangla sign language recognition technique using faster R-CNN(Region-Based Convolutional Neural Network) [16] which is just an extension of the classic CNN used for object detection. A specialized dataset, named BdSLImset, was generated by the researchers to meet their specific research requirements. This paper compares the output accuracy with four other related works which used different image detection algorithms. The mentioned papers obtained accuracies of 96.46%, 98.99%, 97% and 95.80% respectively. Most of the papers that we have reviewed have done some sort of data preprocessing to ensure that the size and quality of the data are being maintained and the performance of the models are optimized. The feature set may also have to be standardized in some cases to properly work with the model in use.

### **2.3.1 Fuzzy Logic**

Using an adaptive fuzzy logic system, the paper [2] proposes a Hand Sign Classification (HSC) system that uses information about the hand motions to classify them as Australian Sign Language. The posture data is classified to recognize the initial and final hand postures as AUSLAN basic hand forms. The start and end postures and the mobility that happened between them are used to classify signs. Both the posture and sign classifications use the same fuzzy procedure, which involves fuzzifying the kinematic input data, rule activation and calculation of matching postures/signs and accompanying likelihood values. The matched rule's output likelihood value indicates the likelihood that the input is the posture or indication that the rule represents.

### **2.3.2 Artificial Neural Network**

The research [12] proposed a Bangla Sign Language Recognition system that trained an artificial neural network using photographs of fingertip location. This method employed position vectors to train an artificial neural network to recognize the relative tip positions of five fingers in a two-dimensional space. Although ANN takes a substantial amount of time and resources to train, it was tested on a custom dataset of 518 photos of 37 indicators and obtained 99% classification accuracy. The authors also offer a method for identifying static alphabet hand gestures in BdSL in [8], which involves training an ANN with sign alphabet features using a feed-forward back-propagation learning algorithm. This method requires photographs of the bare hand for recognition, therefore no gloves or visual marking systems are required. It had a test accuracy of 80.902% on average. In [7], a unique system is proposed

involving Neural Network Ensemble(NNE) where multiple neural networks are combined to form a large network which was used as feature extractor and classifier to detect and classify Bangla sign languages with an accuracy rate of 93%.

### 2.3.3 Principal Component Analysis

Using Principal Component Analysis, the research [10] presents a real-time approach for recognizing and classifying ten English hand sign alphabets (PCA). As patterns in data can be difficult to identify in high-dimensional data, a graphical representation is not an option. PCA can be considered as a useful tool for analyzing such data since it can be used to reduce the dimensionality of the feature space in order to uncover the often hidden, simplified structures that lie behind it without much loss of important information. A video clip collected static frames of hand motions at 3 frames per second. Images in a database containing photographs in the same lighting situation and setting were matched using PCA. In a real-time scenario, the system attained 90% accuracy. Another paper [5] also used PCA to classify 6 Bengali vowels and 10 Bengali numeric characters from sign language to its corresponding written symbol, which amounts to a total of 16 different hand gestures. They preprocessed the original RGB images into HSV color space, taken from a video camera. Afterwards, the images were converted to a binary(black and white) depending on the range of Hue value within the image. These binary images were again converted into grayscale, which were then fed as input to the recognizer. They achieved an average precision and recall rate of 70.5% and 68% for Bengali vowels and 83.7% and 83.6% respectively for Bengali numbers.

### 2.3.4 Hidden Markov Model

The various techniques range from using the Kinect sensor and its various depth-mapping capabilities to extract 20 features from the feed and train a 4-state Hidden Markov Model that is then used to determine the phrase that is implied by a specific gesture. This model is one of the early attempts to recognize sign languages and at that same time, much other research was done on other sign languages. This approach could achieve an accuracy of 51.5% and 76.12% while sitting or standing. Some of the other drawbacks also include the need for proprietary hardware, being the Kinect, to enable this approach [6]. The paper [3] presents a method for continuous American Sign Language (ASL) recognition that uses three-dimensional arm motion data as input. To obtain accurate three-dimensional movement parameters of ASL sentences selected from a 53-sign vocabulary. Afterwards, these parameters were used as features for HMM to detect hand gestures and classify them as ASL.

### 2.3.5 K-Nearest Neighbor

The research in [9] describes a system that recognizes Bengali Sign Language (BdSL) in real-time using computer vision. The system recognizes a possible area of interest from the collected image from a camera, a hand in this case. In order to identify the hand in each frame, the system employs Haar-like feature-based cascaded classifiers. The hand sign is located and classified from the identified hand region based on Hue and Saturation values matching human skin color. The hand sign is converted



to a binary picture after normalization. The binary pictures are then categorized using the K-Nearest Neighbors (KNN) Classifier and compared to pre-trained binary images of hand signs. The system recognizes 6 Bengali vowels and 30 Bengali consonants. The system is trained using 3600 training photographs, then is tested with additional 3600 photos. Vowel identification accuracy was 98.17 percent, while consonant recognition accuracy was 94.75 percent.

### 2.3.6 Support Vector Machine

In [13], the authors provided a framework for utilizing Support Vector Machine to recognize Bangla Sign Language (BSL). The recognition system was trained and tested using the Bangla hand sign alphabets. Bangla sign alphabets are identified by comparing and studying the elements that distinguish each sign. Hand signs are translated to HSV color space from RGB images. Afterwards, the appropriate hand sign characteristics are then acquired using Gabor filters. Since the Gabor filter produces a high-dimensional feature vector, Kernel PCA was utilized to lower the dimensionality without losing much valuable information. Finally, two big SVM classifiers were used for vowel and consonant categorization, respectively. Two different datasets were used for training and testing purposes, respectively. The first dataset contained 2400 images on which the SVM classifier was trained upon. The second dataset also contained 2400 images and these were used to evaluate the model's performance before deploying it. They successfully achieved an accuracy rate of 97.7%.

### 2.3.7 Convolutional Neural Network

The papers [15], [20], [18], [25] and [19] all used a Convolutional Neural Network-based approach to solve the problem of detecting and classifying Bengali sign languages. [18] and [15] are classical CNN approaches where the input image is pre-processed and fed directly to the CNN model for feature extraction. After feature extraction, the features are fed into a neural network that acts as a classifier to distinguish between different sign languages. [18] and [20] used a custom-made model, in which the number of filters and layers of the CNN model is set manually. However, [19] uses a widely known pre-trained model known as VGG19, which was trained on the ImageNet dataset. In contrast, [15] proposed a hybrid approach that involves using Leap Motion Controller (LMC) for detecting hand gestures instead of using raw images. Afterwards, the frames from the LMC are segmented from the time series using Hidden Markov Model. Lastly, the segmented images were fed into a custom-built CNN model to classify them into Bengali sign languages.

### 2.3.8 Contour Analysis

This study [11] uses contour analysis to demonstrate a computer vision-based Bengali sign word recognition system. A Haar-like feature-based cascaded classifier is utilized to locate the predetermined hand posture from the collected image from a camera, which is confined by a bounding box that is initialized as an area of interest (ROI). The method then separates skin-like regions from the ROI using Hue and Saturation values from the normalized image. The system then removes noises using

morphological procedures and Gaussian smoothing before converting the picture to grayscale. The system detects sign words based on the maximum relevance of the extracted contours. The system was trained and evaluated using 1800 contour templates from 10 signers for 18 Bengali sign words, with an accuracy rate of 90.11% and a computational cost of 26.063 milliseconds per frame.

# Chapter 3

## Methodology and System Implementation

### 3.1 Dataset

#### 3.1.1 Dataset Collection

For our intents, we have decided to use the IsharaLipi dataset and will therefore analyze the characteristics of this dataset. From figure 3.1, it is clear that the original dataset is relatively small and is not enough for a deep learning model to learn and generalize necessary features. However, some desirable characteristics of this dataset include the balance it provides and the variability in its background.

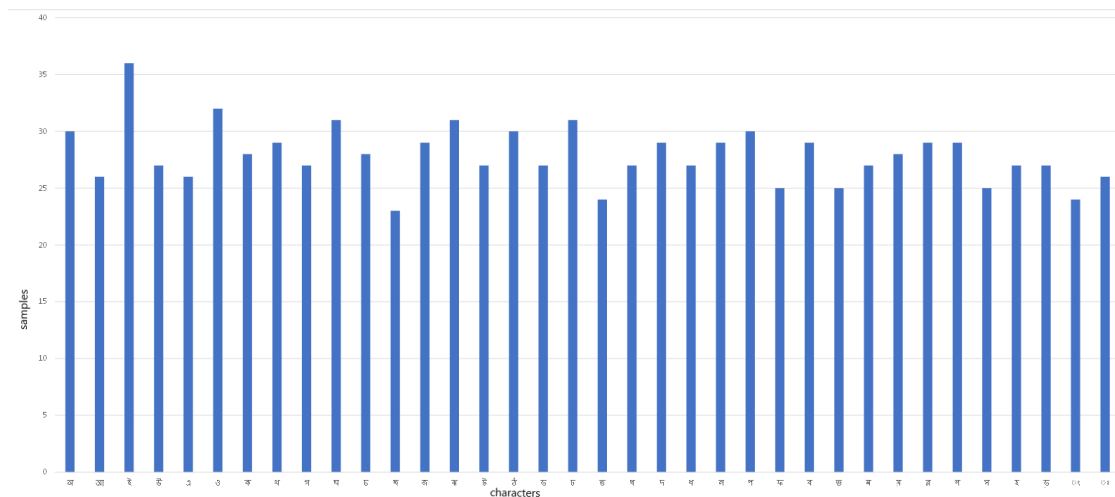


Figure 3.1: Original Samples from IsharaLipi Dataset

From figure 3.2 we can take a glance at each sample from the dataset and conclude that each sign language gesture present in it has been under varying environments and features such as different lighting, angle, skin color, background and clothing. This variability ensures that our model can generalize the key features from the images, which are the hands' orientation, position and overlap.



Figure 3.2: One sample from each class of the original dataset

### 3.1.2 Data Preprocessing

As mentioned in the previous section, the dataset in use is not robust enough for our use cases due to its small average sample size of 20 per class. Therefore, we must resort to various data augmentation techniques to increase the size of our dataset and change the nature of the images so our model may be better equipped to learn only the necessary features from the image.

For this, we used the Augmentor Python library that can take images as input and output them by applying various types of image manipulation algorithms on them. This includes randomly changing the brightness, contrast and colors of the image while also warping the image in certain areas causing random distortions with a probability of 0.4. Moving on we also applied a few transformations on the image such as rotation, shear, skew and zoom with a probability of 0.2, keeping in mind that transforming the images too much may take away from the semantic meaning of the image and cause the model to underperform.

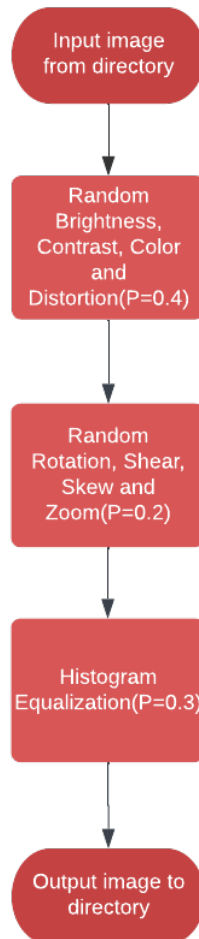


Figure 3.3: Data Augmentation Workflow to produce 1000 samples of images per class where P is the probability of an augment operation

Lastly, we applied a uniform histogram equalization with a probability of 0.3, which is a technique to make the edges of the image crisper that may assist the model when detecting edges.

This resulted in a uniform dataset of 1000 images for each class, as seen in figure 3.4 which is much better than the previously available original dataset. Before feeding this augmented dataset into our model for training, we made sure to convert each image into greyscale images resulting in the images seen in figure 3.5 for each class.

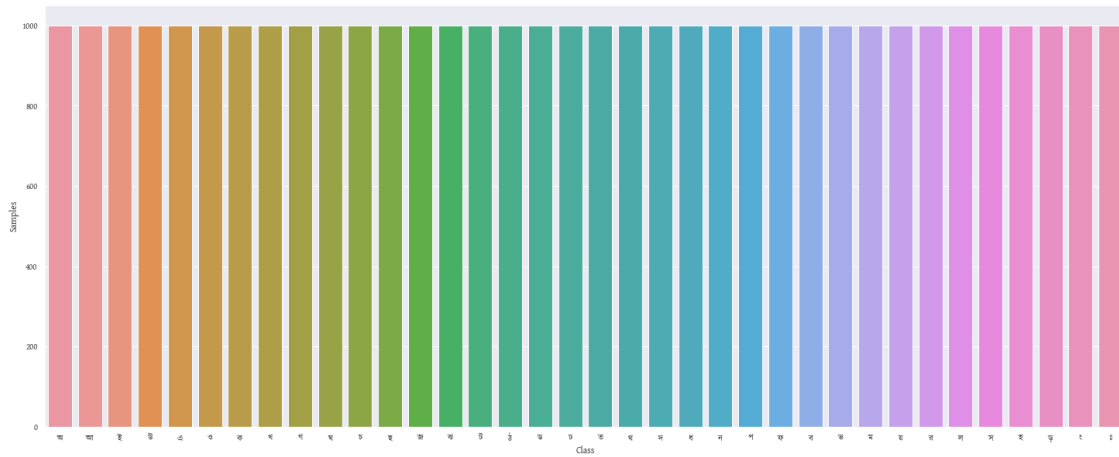


Figure 3.4: Class distribution of augmented samples



Figure 3.5: One sample from each class of the augmented dataset

## 3.2 Model Overview

### 3.2.1 Intuition Behind the Model

Here we discuss the model architecture that was used as the primary method of recognizing the gestures of the Bangla sign language present in our dataset.

The model architecture that we propose in this paper came with a few goals in mind involving accuracy, efficiency and its number of parameters. Most importantly, we aimed for our model to be exceptionally accurate in giving predictions in a wide array of environments. Over here, the environment may include lighting, angle, orientation and other objects. Secondly, our model has a very low number of parameters making it light and responsive when making these predictions. This will ensure that the model can run on low-end devices, including mobile phones, IoT devices, microcontrollers, and web applications(i.e., Zoom or Google Meet) while maintaining precise inferences with lower power consumption.

In order to achieve our goals for this sign language detection model, there are some intuitions that come from knowing about neural networks. Neural networks generally offer better accuracy when they can train over a large dataset. There is no exact number as to how much data a neural network may actually need. However, when a neural network is inferring on a wider range of classes, it must have a larger training data on each class to make meaningful features and differentiate between the classes. Moreover, to ensure that the model is lightweight and fast, the overall architecture of the model must be built with a lower number of parameters that the prediction depends on.

### 3.2.2 Model Architecture

To ensure our primary objectives of increased model efficiency and decreased model size, we opted for a CNN model with 4 convolutional layers, along with 3 max-pooling layers for the feature extractor and 2 dense layers with 256 nodes in the hidden layer and 36 output nodes in the output layer. Each convolutional layer also went through a batch normalization layer to reduce the impact that exploding gradients may have on the model, which can lead to a better generalization on the dataset. This architecture resulted our model to consist of only around 75,000 trainable parameters. The layered architecture of the proposed model and its parameter count are illustrated in tables 3.1 and 3.2 respectively.

Let us dive deeper into the hyperparameters of this architecture. We denote the Input Layer as the top and the Output layer as the bottom. Except the last dense layer, all the layers use the Rectified Linear Unit(ReLU) activation function. The last layer, also called the classification layer, uses the softmax activation function. Next, the convolutional layers have a filter number of 32, 48, 68 and 84 going from the top to the bottom, with a kernel size of (2,2) and a stride of 1 for all convolutional layers except the first one which has a kernel size of (3,3). All Max Pooling layers have pool size of (2,2) and a stride of 2. All convolutional layers have 'same' padding except for the third layer from the top which used 'valid' padding. Lastly, we added 3 Dropout layers with probabilities of 0.2, 0.2 and 0.4 from top to bottom respectively.



#	Layer Name	Layer Type	Filters/Nodes	Kernel/Pool Size	Stride	Padding	Output Shape
1	input_1	InputLayer	N/A	N/A	N/A	N/A	(None, 64, 64, 3)
2	conv2d	Conv2D	32	(3, 3)	1	same	(None, 64, 64, 32)
3	batch_normalization	BatchNormalization	N/A	N/A	N/A	N/A	(None, 64, 64, 32)
4	conv2d_1	Conv2D	48	(2, 2)	1	same	(None, 64, 64, 48)
5	max_pooling2d	MaxPooling2D	N/A	(2, 2)	2	valid	(None, 32, 32, 48)
6	batch_normalization_1	BatchNormalization	N/A	N/A	N/A	N/A	(None, 32, 32, 48)
7	conv2d_2	Conv2D	68	(2, 2)	1	valid	(None, 31, 31, 68)
8	batch_normalization_2	BatchNormalization	N/A	N/A	N/A	N/A	(None, 31, 31, 68)
9	max_pooling2d_1	MaxPooling2D	N/A	(2, 2)	2	valid	(None, 15, 15, 68)
10	dropout	Dropout	N/A	N/A	N/A	N/A	(None, 15, 15, 68)
11	conv2d_3	Conv2D	84	(2, 2)	1	same	(None, 15, 15, 84)
12	batch_normalization_3	BatchNormalization	N/A	N/A	N/A	N/A	(None, 15, 15, 84)
13	max_pooling2d_2	MaxPooling2D	N/A	(2, 2)	2	valid	(None, 7, 7, 84)
14	dropout_1	Dropout	N/A	N/A	N/A	N/A	(None, 7, 7, 84)
15	global_average_pooling2d	GlobalAveragePooling2D	N/A	N/A	N/A	N/A	(None, 84)
16	dense	Dense	256	N/A	N/A	N/A	(None, 256)
17	dropout_2	Dropout	N/A	N/A	N/A	N/A	(None, 256)
18	classification	Dense	36	N/A	N/A	N/A	(None, 36)

Table 3.1: Sequential architecture and hyperparameters of our proposed model

A simplified 3D-layered architecture of our model can also be observed in 3.6 where the dropout layers are typically used in the later parts of the model since this is where the model usually specializes over the dataset, which may lead to overfitting.

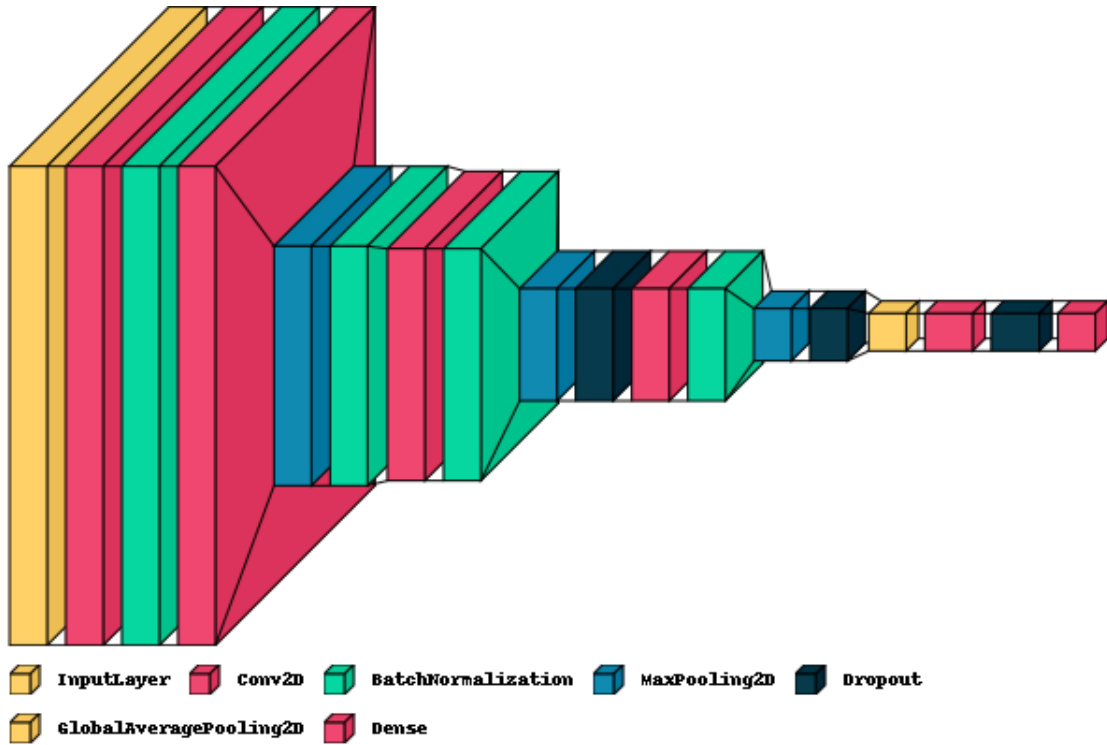


Figure 3.6: Summarized Plot and Propagation from Layer to Layer

### 3.2.3 Model Efficiency Considerations

Now let us go over how we worked on these two aspects of building our model architecture. Firstly, since we are building a model on Bengali Sign Language detection, we had to collect many images of Bengali sign language gestures that were

Total Parameters	75,084
Trainable parameters	74,620
Non-trainable parameters	464

Table 3.2: Parameter count of our proposed model

annotated with the proper classes. We aggregated several datasets that we will be using for our own purposes and later pre-processed all the images using the same technique. Secondly, to make our model lightweight, we built a relatively shallow neural network model with a reduced number of layers and then focused extensively on hyperparameter tuning to increase the accuracy of our model.

Additionally, from a general understanding of the problem of sign language detection, we have recognized that color in images is not a necessity for recognizing the gesture[6]. The features we may need from an image are the outline of a hand, the position of the hands with respect to each other and skin color. The skin color may help identify lighting conditions and depth of field. However, for our purposes, it would not help differentiate it from the background. Therefore, after preprocessing our images, the images are of dimensions 64x64 and in greyscale color scheme.

Along with this, we used data augmentation before the training phase to make our data more robust to any kind of environment. To keep it relevant to real world scenarios, the data augmentation phase will only mutate the images in minor ways by randomly rotating, shearing and zooming the images. These steps will help reduce overfitting as much as possible, which is a major issue when deploying a model in the real world. Notably, we have used histogram equalization to ensure that the outlines of the hands are more prominent.

### 3.3 System Workflow

Now that we have established the base goal for our model we may move onto the implementation. The detailed workflow for our model has been illustrated in figure 3.7 where we clarify each step that goes into building and improving our model.

Firstly, we created a base model keeping in mind the goals we wanted to achieve for this problem. Later, we compiled the dataset with the Adam optimizer and a learning rate of 0.001. After training for 10 epochs, we evaluate whether the model is overfitting or not by observing the validation accuracy and loss.

After that, when we reach a high validation accuracy with our model, that is over 99%, we can experiment and tune the hyperparameters so it may achieve similar results with the testing set as well. In this stage, it is critical to ensure that the model does not overfit on validation data which is quite possible if the model is tuned according to it.

Lastly, the model may be deployed and used in practice as it is proven to work with acceptable accuracy over the various subsets of the dataset.

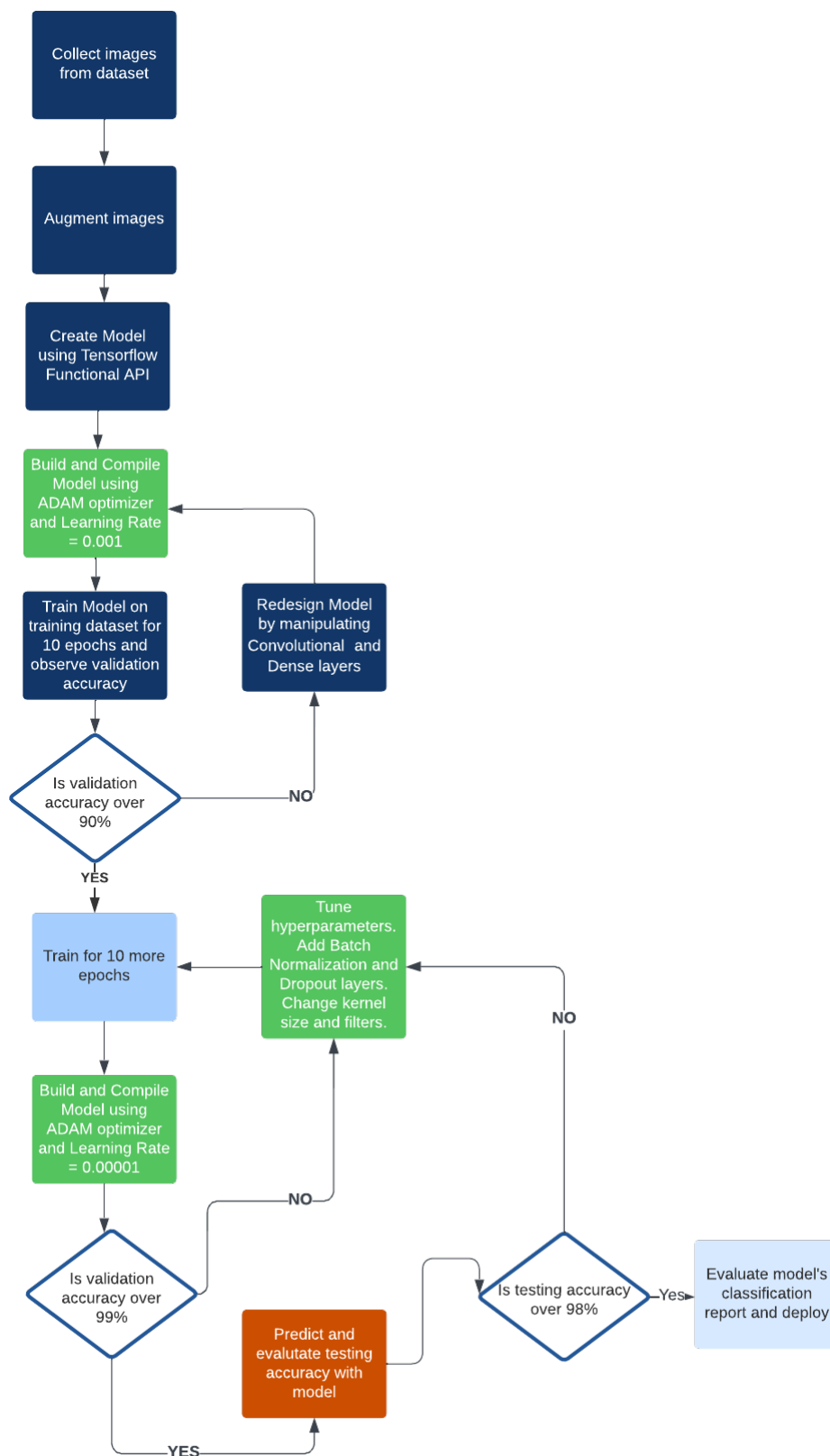


Figure 3.7: Workflow of creating, tuning and building our custom model

## 3.4 Model Training and Testing

### 3.4.1 Overfitting & Under fitting

During our experimentation, our results pointed towards one major problem when working small datasets that is overfitting. Primarily, when working with CNNs the process of feature extraction is effective enough to learn the underlying features from images; however, they are not always generalized to the problem at hand. To alleviate this, we have used various techniques such as regularization and some specific callback functions when training our model.

#### Regularization

The primary regularization methods that may be used are L1 Regularization, L2 Regularization and using Dropout Layers. L1 and L2 regularization help curb overfitting by penalizing input weights that contribute very highly towards the overall activation of a node. Dropout layers randomly drop inputs to a node, forcing the model to extract only the important features from a dataset, leading to better generalization.[24]

#### Callbacks

The main callback functions we used were Early Stopping and Learning Rate Reduction. Early Stopping helps stop model training before the model overfits by halting training when the validation accuracy and training accuracy start diverging. Learning Rate Reduction reduces the learning rate when the training loss plateaus to a global or local minima value so that it may not be overshoot the minima loss value, leading to decreased validation accuracy.

## 3.5 Model Deployment Considerations

As discussed before, the model must be lightweight in order to be accessible to a wider range of people. However, it is not practical or efficient to host the entire model on the consumer end device. Instead, these models will be hosted and deployed on cloud platforms such as AWS and Heroku, where they will be called via an API to process any data and give meaningful inferences. This workflow makes the most sense if we want close to real-time performance for mobile devices. Of course, this brings the limitations that the device must always be connected to the internet.

# Chapter 4

## Experimental Results and Analysis

### 4.1 Experimental Results and Analysis

From the previous analysis of alternative models and problems, we could extrapolate that the major drawback of building a model for sign language recognition was the scarcity of datasets that cater to this particular problem. However, using regularization and data augmentation techniques, we overcame this issue and developed a model that acquired 99% validation accuracy on our augmented dataset.

### 4.2 Model Accuracy and Loss

From figures 4.1 and 4.2 we can clearly see that the model slowly transitions from a low accuracy to a high accuracy over the span of 10 epochs. The disparity between the validation and training accuracy/loss can be attributed to the Batch Normalization and Dropout layers being inactive during validation.

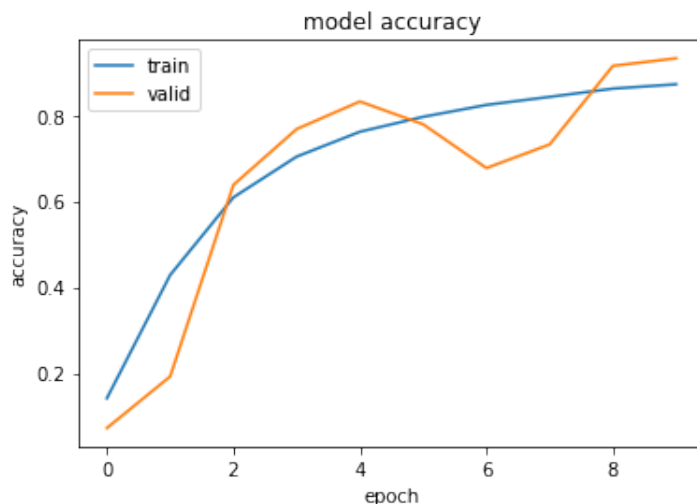


Figure 4.1: Transition of Model Accuracy over 10 epochs

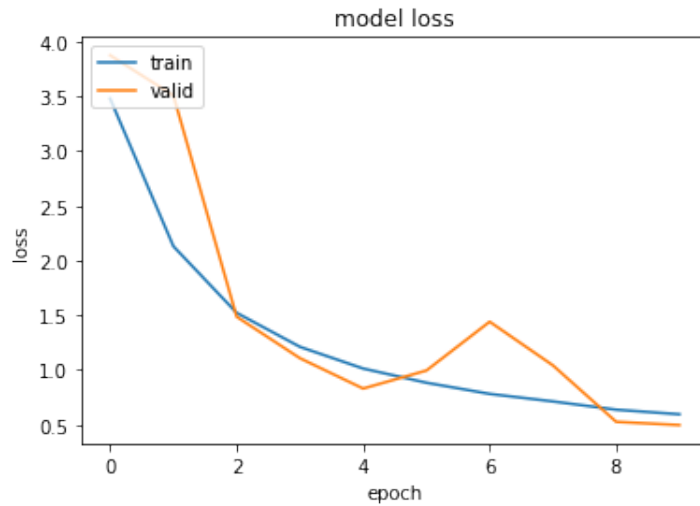


Figure 4.2: Transition of Model Loss over 10 epochs using Categorical Crossentropy

## 4.3 Model Evaluation

This section will highlight the methods we used to further evaluate the performance of our model.

### 4.3.1 Confusion Matrix

The confusion matrix in figure 4.3 gives us a clear picture of how many images were correctly classified, denoted by the diagonal of the matrix. We can also see the ones that were incorrectly classified by looking at the matrix cells outside of the diagonal. For this specific problem, the confusion matrix may not give us enough context to explain what the model got wrong since sign labeled as 5 may or may not be similar to sign labeled as 6.

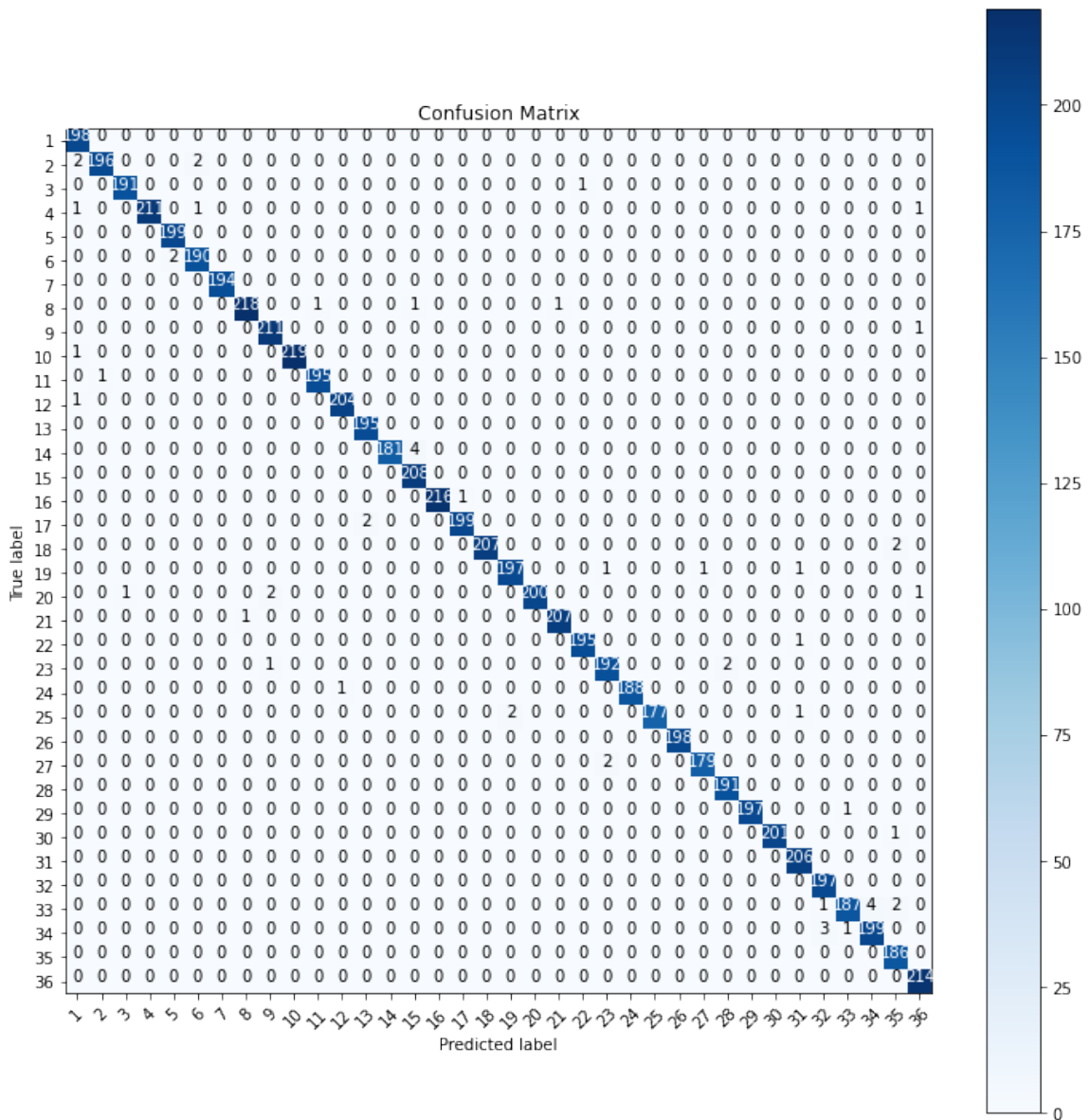


Figure 4.3: Confusion Matrix for our Custom Model

### 4.3.2 Other Metrics

For the other metrics, we have used recall, precision, f1-score and support as a measure of how well the model has performed over the whole dataset. From table 4.1, we can have a very clear representation of all the metrics for each class of our dataset. From the average precision and recall, we can say that the model works well in avoiding false positives as well as false negatives. It is also evident that our model has a high mean value of precision and recall, indicating our model is robust and accurate enough to detect and classify all the different hand gestures of the Bangladeshi sign language.

Class	Precision	Recall	f1-score	Support
0	0.98	1.00	0.99	198
1	0.99	0.98	0.99	200
2	0.99	0.99	0.99	192
3	1.00	0.99	0.99	214
4	0.99	1.00	0.99	199
5	0.98	0.99	0.99	192
6	1.00	1.00	1.00	194
7	1.00	0.99	0.99	221
8	0.99	1.00	0.99	212
9	1.00	1.00	1.00	220
10	0.99	0.99	0.99	196
11	1.00	1.00	1.00	205
12	0.99	1.00	0.99	195
13	1.00	0.98	0.99	185
14	0.98	1.00	0.99	208
15	1.00	1.00	1.00	217
16	0.99	0.99	0.99	201
17	1.00	0.99	1.00	209
18	0.99	0.98	0.99	200
19	1.00	0.98	0.99	204
20	1.00	1.00	1.00	208
21	0.99	0.99	0.99	196
22	0.98	0.98	0.98	195
23	1.00	0.99	1.00	189
24	1.00	0.98	0.99	180
25	1.00	1.00	1.00	198
26	0.99	0.99	0.99	181
27	0.99	1.00	0.99	191
28	1.00	0.99	1.00	198
29	1.00	1.00	1.00	202
30	0.99	1.00	0.99	206
31	0.98	1.00	0.99	197
32	0.99	0.96	0.98	194
33	0.98	0.98	0.98	203
34	0.97	1.00	0.99	186
35	0.99	1.00	0.99	214
accuracy			0.99	7200
macro avg	0.99	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99	0.99

Table 4.1: Classification Report on Testing Dataset



# Chapter 5

## Conclusion

### 5.1 Conclusion

We present a system that combines deep learning with computer vision methods to recognize and categorize Bangla sign languages. With a test accuracy of 99.21%, our custom-built CNN model can detect and categorize Bangla sign language symbols from the Ishara-Lipi dataset. We trained our model with enough samples by enriching the Ishara-Lipi dataset using various data augmentation techniques to distinguish the specific signals of a hand gesture and grasp what they signify. Our model is a classical CNN approach based on the vanilla CNN architecture, which is flexible enough to work in a variety of scenarios, as we plan to train the model on other datasets, potentially improving the validity accuracy. More research on this topic could lead to the detection of complete phrases from gestures detected by similar models.

### 5.2 Practical Implications

From this work, we are very specifically working on the character classification part of the problem when approaching this paper. Our model is designed to classify single or double-hand Bengali Sign Language gestures and output the corresponding letter. This solution may be used to facilitate environments where there is a lack of proper instruments for sign language users to communicate. This may also be used in education to train all people the skill of sign language without the intervention of a teacher. Lastly, this may be paired with technologies akin to stenography to create keyboard-less environments for sign language users so they may communicate easily using phonetic language and word prediction rather than spelling out entire words using characters.

### 5.3 Future Work

For future work, we hope to see our model being used in the final pipe when classifying static Bengali Sign Language characters after it has been boxed using various object detection methods. We may also use NLP models to string together characters and from words on the fly, using the character output from our model. More work needs to be done to recognize individual words and sentences using moving

hands, bodies, and faces in the sign language realm. The release of a dataset containing video files of these moving gestures and relevant body parts may be the first step to achieving this feat.

# Bibliography

- [1] B. d. Gelder, J. Vroomen, and L. van der Heide, “Face recognition and lip-reading in autism,” en, *European Journal of Cognitive Psychology*, vol. 3, no. 1, pp. 69–86, Jan. 1991, ISSN: 0954-1446, 1464-0635. DOI: 10.1080/09541449108406220. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/09541449108406220>.
- [2] Eun-Jung Holden, G. Roy, and R. Owens, “Adaptive classification of hand movement,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 3, Perth, WA, Australia: IEEE, 1995, pp. 1373–1378, ISBN: 9780780327689. DOI: 10.1109/ICNN.1995.487358. [Online]. Available: <http://ieeexplore.ieee.org/document/487358/>.
- [3] C. Vogler and D. Metaxas, “Adapting hidden Markov models for ASL recognition by using three-dimensional computer vision methods,” in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 1, Orlando, FL, USA: IEEE, 1997, pp. 156–161, ISBN: 9780780340534. DOI: 10.1109/ICSMC.1997.625741. [Online]. Available: <http://ieeexplore.ieee.org/document/625741/>.
- [4] S. J. F. Fudickar and K. Nurzynska, “A User-Friendly Sign Language Chat,” in *Conference ICL2007, September 26 -28, 2007*, M. E. Auer, Ed., Villach, Austria: Kassel University Press, 2007, 7 pages. [Online]. Available: <https://telearn.archives-ouvertes.fr/hal-00197225>.
- [5] S. Begum and M. Hasanuzzaman, “Computer vision-based bangladeshi sign language recognition system,” in *2009 12th International Conference on Computers and Information Technology*, IEEE, 2009, pp. 414–419.
- [6] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti, “American sign language recognition with the kinect,” en, in *Proceedings of the 13th international conference on multimodal interfaces - ICMI ’11*, Alicante, Spain: ACM Press, 2011, p. 279, ISBN: 9781450306416. DOI: 10.1145/2070481.2070532. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2070481.2070532>.
- [7] B. ChandraKarmokar, K. Md. Rokibul Alam, and M. Kibria Siddiquee, “Bangladeshi Sign Language Recognition Employing Neural Network Ensemble,” *International Journal of Computer Applications*, vol. 58, no. 16, pp. 43–46, Nov. 2012, ISSN: 09758887. DOI: 10.5120/9370-3846. [Online]. Available: <http://research.ijcaonline.org/volume58/number16/pxc3883846.pdf>.
- [8] M. A. Rahman, “Recognition of Static Hand Gestures of Alphabet in Bangla Sign Language,” *IOSR Journal of Computer Engineering*, vol. 8, no. 1, pp. 07–13, 2012, ISSN: 22788727, 22780661. DOI: 10.9790/0661/0810713. [Online].

Available: <http://www.iosrjournals.org/iosr-jce/papers/Vol8-Issue1/B0810713.pdf>.

- [9] M. A. Rahaman, M. Jasim, M. H. Ali, and M. Hasanuzzaman, "Real-time computer vision-based Bengali Sign Language recognition," in *2014 17th International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh: IEEE, Dec. 2014, pp. 192–197, ISBN: 9781479962884. DOI: 10.1109/ICCITechn.2014.7073150. [Online]. Available: <http://ieeexplore.ieee.org/document/7073150/>.
- [10] A. Saxena, D. K. Jain, and A. Singhal, "Sign language recognition using principal component analysis," in *2014 Fourth International Conference on Communication Systems and Network Technologies*, IEEE, 2014, pp. 810–813.
- [11] M. A. Rahaman, M. Jasim, M. H. Ali, and M. Hasanuzzaman, "Computer vision based Bengali sign words recognition using contour analysis," in *2015 18th International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh: IEEE, Dec. 2015, pp. 335–340, ISBN: 9781467399302. DOI: 10.1109/ICCITechn.2015.7488092. [Online]. Available: <http://ieeexplore.ieee.org/document/7488092/>.
- [12] 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)*, Dhaka, Bangladesh: IEEE, Dec. 2016, pp. 1–5, ISBN: 9781509054213. DOI: 10.1109/MEDITEC.2016.7835364. [Online]. Available: <http://ieeexplore.ieee.org/document/7835364/>.
- [13] M. A. Uddin and S. A. Chowdhury, "Hand sign language recognition for Bangla alphabet using Support Vector Machine," in *2016 International Conference on Innovations in Science, Engineering and Technology (ICISSET)*, Dhaka, Bangladesh: IEEE, Oct. 2016, pp. 1–4, ISBN: 9781509061228. DOI: 10.1109/ICISSET.2016.7856479. [Online]. Available: <http://ieeexplore.ieee.org/document/7856479/>.
- [14] M. D Bloice, C. Stocker, and A. Holzinger, "Augmentor: An Image Augmentation Library for Machine Learning," *The Journal of Open Source Software*, vol. 2, no. 19, p. 432, Nov. 2017, ISSN: 2475-9066. DOI: 10.21105/joss.00432. [Online]. Available: <http://joss.theoj.org/papers/10.21105/joss.00432>.
- [15] F. Yasir, P. W. C. Prasad, A. Alsadoon, A. Elchouemi, and S. Sreedharan, "Bangla Sign Language recognition using convolutional neural network," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kerala State, Kannur, India: IEEE, Jul. 2017, pp. 49–53, ISBN: 9781509061068. DOI: 10.1109/ICICICT1.2017.8342533. [Online]. Available: <http://ieeexplore.ieee.org/document/8342533/>.
- [16] O. B. Hoque, M. I. Jubair, M. S. Islam, A.-F. Akash, and A. S. Paulson, "Real Time Bangladeshi Sign Language Detection using Faster R-CNN," in *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, Dhaka, Bangladesh: IEEE, Dec. 2018, pp. 1–6, ISBN: 9781538652299. DOI: 10.1109/ICIET.2018.8660780. [Online]. Available: <https://ieeexplore.ieee.org/document/8660780/>.

- [17] M. Sanzidul Islam, S. Sultana Sharmin Mousumi, N. A. Jessan, A. Shahariar Azad Rabby, and S. Akhter Hossain, “Ishara-Lipi: The First Complete Multipurpose Open Access Dataset of Isolated Characters for Bangla Sign Language,” in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sylhet: IEEE, Sep. 2018, pp. 1–4, ISBN: 9781538682074. DOI: 10.1109/ICBSLP.2018.8554466. [Online]. Available: <https://ieeexplore.ieee.org/document/8554466/>.
- [18] M. S. Islalm, M. M. Rahman, M. H. Rahman, M. Arifuzzaman, R. Sassi, and M. Aktaruzzaman, “Recognition Bangla Sign Language using Convolutional Neural Network,” in *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, Sakhier, Bahrain: IEEE, Sep. 2019, pp. 1–6, ISBN: 9781728130125. DOI: 10.1109/3ICT.2019.8910301. [Online]. Available: <https://ieeexplore.ieee.org/document/8910301/>.
- [19] A. M. Rafi, N. Nawal, N. S. N. Bayev, L. Nima, C. Shahnaz, and S. A. Fattah, “Image-based Bengali Sign Language Alphabet Recognition for Deaf and Dumb Community,” in *2019 IEEE Global Humanitarian Technology Conference (GHTC)*, Seattle, WA, USA: IEEE, Oct. 2019, pp. 1–7, ISBN: 9781728117805. DOI: 10.1109/GHTC46095.2019.9033031. [Online]. Available: <https://ieeexplore.ieee.org/document/9033031/>.
- [20] L. K. S. Tolentino, R. O. S. Juan, A. C. Thio-ac, M. A. B. Pamahoy, J. R. R. Forteza, and X. J. O. Garcia, “Static sign language recognition using deep learning,” *International Journal of Machine Learning and Computing*, vol. 9, no. 6, pp. 821–827, 2019.
- [21] O. B. Hoque, M. I. Jubair, A.-F. Akash, and S. Islam, “Bdsl36: A dataset for bangladeshi sign letters recognition,” in *Proceedings of the Asian Conference on Computer Vision*, 2020.
- [22] Abdullah Al Jaid Jim, *KU-BdSL: Khulna University Bengali Sign Language dataset*, Type: dataset, Sep. 2021. DOI: 10.17632/SCPVM2NBKM.1. [Online]. Available: <https://data.mendeley.com/datasets/scpvm2nbkm/1>.
- [23] A. Khatun, M. S. Shahriar, M. H. Hasan, K. Das, S. Ahmed, and M. S. Islam, “A Systematic Review on the Chronological Development of Bangla Sign Language Recognition Systems,” in *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Kitakyushu, Japan: IEEE, Aug. 2021, pp. 1–9, ISBN: 9781665449236. DOI: 10.1109/ICIEVicIVPR52578.2021.9564157. [Online]. Available: <https://ieeexplore.ieee.org/document/9564157/>.
- [24] M. Al-Qurishi, T. Khalid, and R. Souissi, “Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues,” *IEEE Access*, vol. 9, pp. 126 917–126 951, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3110912. [Online]. Available: <https://ieeexplore.ieee.org/document/9530569/>.

- [25] A. S. M. Miah, J. Shin, M. A. M. Hasan, and M. A. Rahim, “BenSignNet: Bengali Sign Language Alphabet Recognition Using Concatenated Segmentation and Convolutional Neural Network,” en, *Applied Sciences*, vol. 12, no. 8, p. 3933, Apr. 2022, ISSN: 2076-3417. DOI: 10.3390/app12083933. [Online]. Available: <https://www.mdpi.com/2076-3417/12/8/3933>.
- [26] *BdSL-D1500*, en. [Online]. Available: <https://www.kaggle.com/kanchonkantipodder/bdsl-d1500>.
- [27] *Deaf in America — Carol Padden, Tom Humphries*, en. [Online]. Available: <https://www.hup.harvard.edu/catalog.php?isbn=9780674194243>.
- [28] *National Activities - Bangladesh. - Bangladesh Sign language Day*. [Online]. Available: <http://www.dpiap.org/national/article.php?countryid=017&id=0000029&country=Bangladesh..>