Detecting Letters and Words from Bangladeshi Sign Language in Real-time

A thesis
submitted in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Submitted by

Oishee Bintey Hoque	**150104005**
Al-Farabi Akash	**150104010**
Md. Saiful Islam	**150104027**
Alvin Sachie Paulson	**150104035**

Supervised by

Mr. Mohammad Imrul Jubair
Assistant Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dhaka, Bangladesh

June 2019

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Mr. Mohammad Imrul Jubair, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis-I and CSE4250: Project and Thesis-II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Oishee Bintey Hoque
150104005

Al-Farabi Akash
150104010

Md. Saiful Islam
150104027

Alvin Sachie Paulson
150104035

CERTIFICATION

This thesis titled, **“Detecting Letters and Words from Bangladeshi Sign Language in Real-time”**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in June 2019.

Group Members:

Oishee Bintey Hoque	**150104005**
Al-Farabi Akash	**150104010**
Md. Saiful Islam	**150104027**
Alvin Sachie Paulson	**150104035**

Mr. Mohammad Imrul Jubair
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Prof. Dr. Kazi A. Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

The culmination of our thesis is a great accomplishment for us. It has been a great journey from selecting the topic to finally taking the project to a level where the research holds great promise. We got the opportunity to learn a lot throughout the course of our research. We would first like to thank our supervisor, Mohammad Imrul Jubair, for his continuous support throughout this journey. Without his enthusiasm, motivation and careful guidance, this thesis would never have taken shape. He was always there to help us whenever we ran into a trouble, faced problems with codes or had questions about our research. He always pushed us to take the challenges and do what seemed most challenging, and most of all never lost confidence in us.

We owe a deep sense of gratitude to Prof. Dr. Kazi A. Kalpoma, Head of the department of Computer Science and Technology, for her continuous support and encouragement in our thesis. Also special thanks to our externals for reviewing our thesis book.

We are deeply indebted to the external examiners of our thesis, Faisal Muhammad Shah and Shoeb Mohammad Shahriar, for their informative suggestions. Lastly, it would not be complete without mentioning all the faculty members and other staffs for their enormous support, our parents for their encouragement, support and attention. We thank profusely 'DHAKA BADHIR HIGH SCHOOL' which provided us necessary information for implementing in user level.

Our thanks and appreciation also go to the people who helped directly or indirectly in our thesis.

Dhaka
June 2019

Oishee Bintey Hoque

Al-Farabi Akash

Md. Saiful Islam

Alvin Sachie Paulson

ABSTRACT

Bangladeshi Sign Language (BdSL) is a commonly used medium of communication for the hearing-impaired people in Bangladesh. Sign Language Recognition (SLR) aims at the computer-based model, translating the Sign Language (SL) into speech or text, so as to facilitate the communication between hearing-impaired people and the normal people. This problem has a broad social impact and an interesting avenue of research. However, it is a challenging task due to the variation of different people (age, gender, etc) and the complexity in sign words. Sign word recognition gets more difficult due to variation in action. Developing a real-time system to detect these signs from images or videos is a great challenge. Such type of work is rare, especially no work on sign word recognition has been done so far in Bangladesh. Also developing a system that works for both letters and words is both unique and challenging. In our thesis, we present different methodologies to detect BdSL letters and words from real-time videos as sign letters do not consist of any sequence of actions whereas sign words consist of different actions. One of our methods uses a Convolutional Neural Network based object detection technique to detect the presence of signs in the image region and recognize its class. For this purpose, we adopted Faster Region-based Convolutional Network model in our system. On the other hand, for the sign word recognition, Recurrent Neural Network(RNN) based model has been applied. The videos are converted to multiple frames and then the sequence of images are processed to recognize the class. Both of the methods are implemented on our own generated dataset. There are no such datasets available on BdSL and the sign languages are significantly different in different countries and even regions. So, we had to develop datasets – *BdSLImset* & *BdSLVidset*– by ourselves to train our systems and we have given open access to our datasets for future research. Using deep learning we have got results with high accuracy. To demonstrate the application, we have developed a Graphical User Interface (GUI) for our system which can detect the letters and words respectively in real-time, and give satisfactory output.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sign language is a non-verbal form of communication used especially by people with the inability to speak or hear. They use it to express their thoughts and emotions. But non-signers find it difficult to understand them. We tried to understand the sign language communication, investigated how they actually communicate and made an approach, so that we could create a bridge between the signers and non-signers.

1.1 Overview

The people who cannot speak, communicate through sign language. This form of communication involves varieties of hand movements and gestures as well. As Bangladeshi Sign Language (BdSL) is structurally different from sign languages of other countries, there is huge work to be done to make this way of communication easier. Communication is an essential part of everyday life and it is particularly important for deaf people to communicate as normally as possible with others. However, there is a lack of qualified Sign Language (SL) interpreters (SLI) in Bangladesh and high demand for these type of services. This problem is further aggravated by the small number of general people who understand this form of communication, making Sign Language (SL) communication more difficult. So there is a need to solve this matter as the people who can not speak already face major difficulties in communication.
There is a specific sign for each Bengali letters (see Figure 1.1). We tried to work with each of these letters. BdSL letters are difficult and consist of different gestures. They use the characters to spell anything letter by letter or say their names. Some of the letters also have similar gestures and it is very hard to differentiate these letters for a computer vision system.
The people who can not speak, sometimes use characters for their communication. But

they do not use it often. They tend to use an action to describe a specific word or phrase. Characters do not consist of any motion whereas words have a sequence of motions in the gesture.

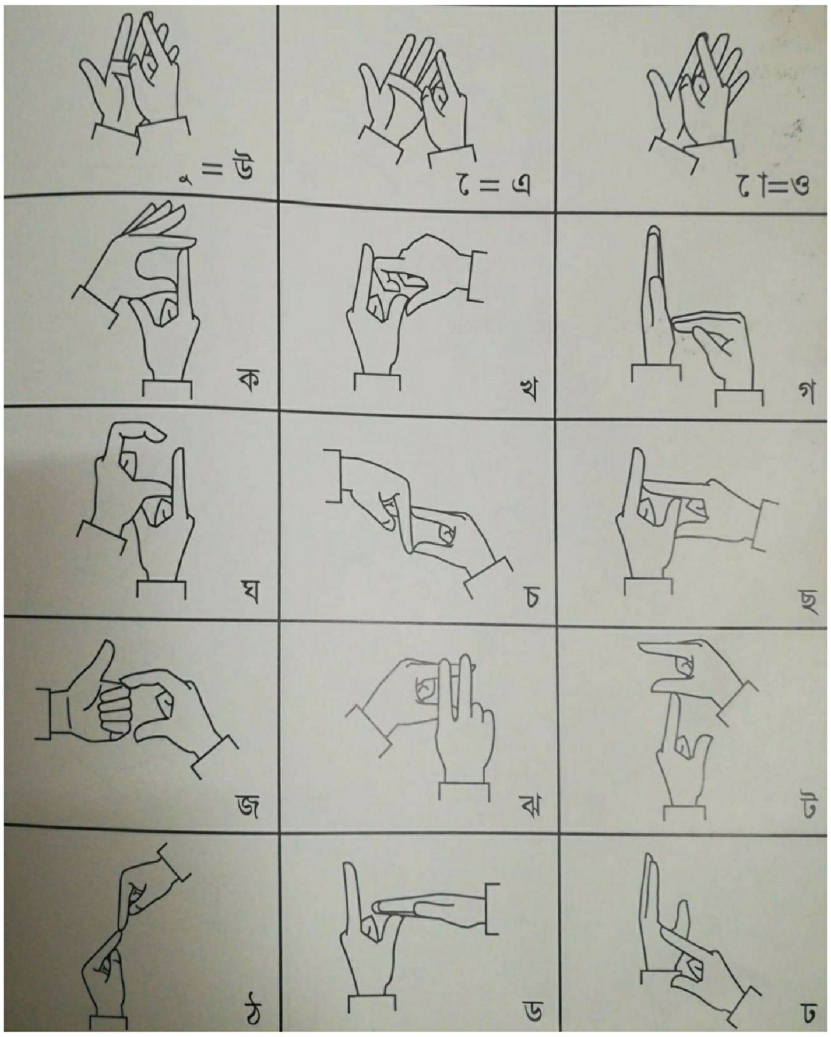


Figure 1.1: A set of Bengali Sign Letters, collected from the book taught in Bodhir School (A school for the hearing impaired people located at Purana Paltan, Dhaka)

1.2 Research Domain

Our main objective is Bangladeshi Sign Language to be perceived and understood by everyone via a digital medium, i.e, from videos or images. First of all, we focused on letter based recognition of sign language which allows the system to track down a single letter which eventually falls under the domain of object detection. Letters are necessary for forming up a single word but mostly each word has its own gesture. But words consist of action, that extends our research domain towards action recognition from temporal data, unlike letters.

Therefore, we came up with a different solution for recognizing sign words in real-time. We briefly describe these two research sub-domain for our thesis in the following sections.

1.2.1 Object Detection and Localization

Object detection and localization is a system related to computer vision and image processing and deals with detecting objects of a certain class from an image and localize where it is present in the image. Every object class has its own special features that help in classifying the class. And by sub-dividing an image into different regions, the system can search if the object is present in that region. For example, we are looking for, what type of objects are present in a picture and where they are in the image. Such an example of multiple object detection is given in Figure 1.2.

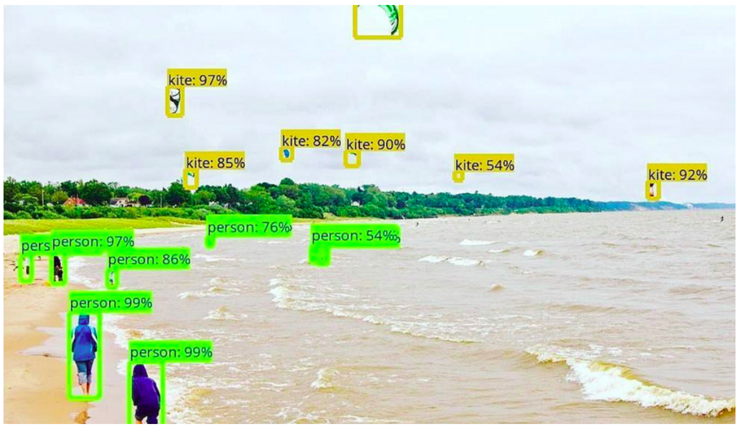


Figure 1.2: Example of Object Detection [1]

As sign letters have no sequence of motion or action, recognition of sign letters is similar to object detection and localization methods. Each sign letter can be considered as an individual object. Most of the time object (letter) recognition problem consists of three tasks to be done in real time:

1. Obtaining a video of the user signing a letter- **Input**.
2. Identifying if there is any letter in the video frame- **Classification**.
3. Reconstructing and displaying a bounding box on the gesture with the most likely letter from classification scores- **Output**.

From a computer vision perspective, this problem represents a significant challenge due to a number of considerations, including:

1. Environmental concerns (e.g. lighting sensitivity, background, and camera position).

2. Occlusion (e.g. some or all fingers, or an entire hand can be out of the field of view).
3. Sign boundary detection (when a sign ends and the next begins).
4. Co-articulation (when a sign is affected by the preceding or succeeding sign).

To overcome the barriers, the simplest deep learning approach, and a widely used one, for detecting objects in images is- Neural Network (Figure 1.3) and Convolutional Neural Networks (CNN) (Figure 1.4).

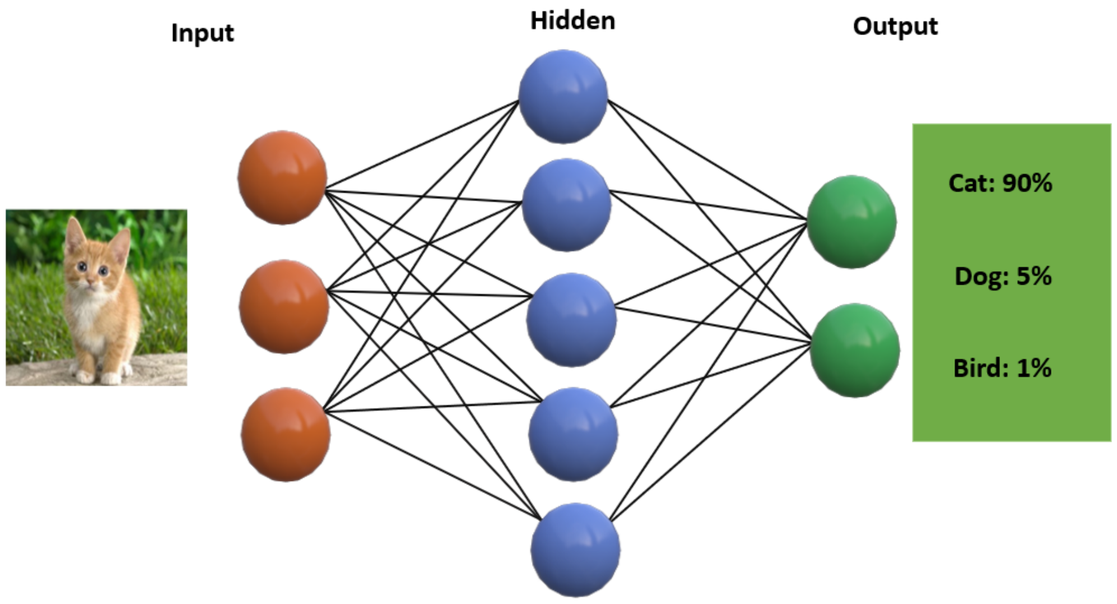


Figure 1.3: Basic Neural Network Model. Image values are flattened in a 1D vector and goes through some hidden layers with some initial weights. Weights are updated during training through forward and backward pass and the system learns to classify objects from images.

There are some improvised region based CNN (R-CNN) methods that are helpful in the localization of objects in an image. Faster R-CNN, one such model proposes a bunch of boxes in the image and checks if any of these boxes contain any object. This R-CNN model uses selective search to extract these boxes from an image (these boxes are called regions), shown in Figure 1.5.

Another R-CNN based models such as, You Only Look Once (YOLO) which makes predictions with a single network evaluation unlike systems like other R-CNNs which require more regional computation for a single image, shown in Figure 1.6. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

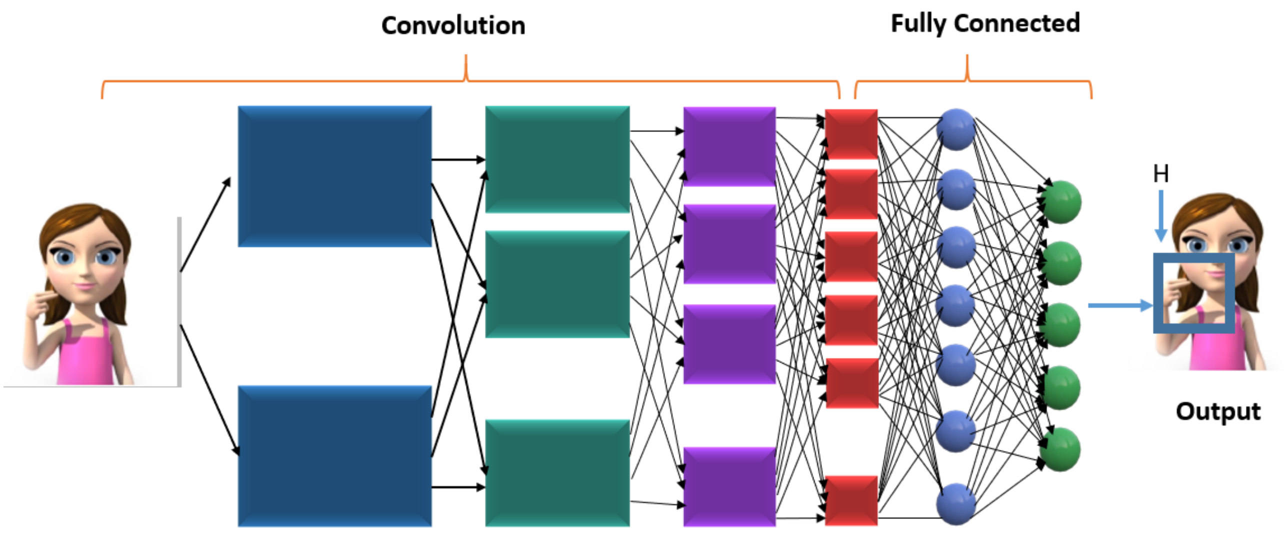


Figure 1.4: Basic CNN Model. Before flattening the image values, these values go through some convolutional layers and then fed into a neural network to train.

1.2.2 Action Recognition

Activity recognition aims to recognize the actions and predicts one or more item from a series of observations on the item's actions and the environmental conditions. As mentioned earlier, sign words contain a sequence of actions. This problem falls under the domain of action recognition in deep learning.

Unlike object detection, the system has to combine the sequence of images/frames to identify a class. Recurrent Neural Network (RNN) is a very popular sequence model, which combines a sequence of inputs and trains the system. As each frame is dependent on the frames, by sharing the parameters among each other RNN can learn these sequences. Long Short Term Memory (LSTM) is an RNN based model that keeps track of even very earlier frames outputs. There can be different types of RNN models, such as one to one, one to many, many to many, many to one. Sign word recognition falls under many to one RNN model. As many frames go as input and output is one class prediction (see Figure 1.7).

1.2.3 Challenges

Making such a system was not smooth. We faced many challenges. To solve these problems, the first thing we needed was a good understanding of how Bangladeshi Sign Language works. We have learned the signs, researching the signs from several sources. Based on that we had to make our own dataset as there was no suitable dataset available to work with. Another problem we faced was that very few works have been done using deep learning in

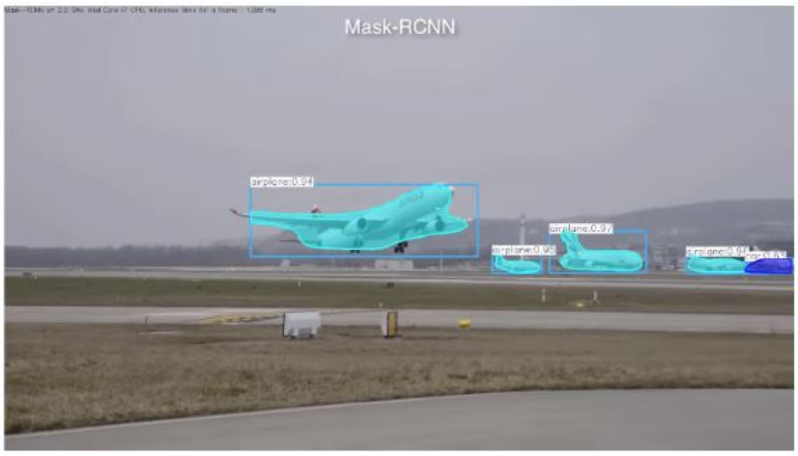


Figure 1.5: R-CNN Masking [2]

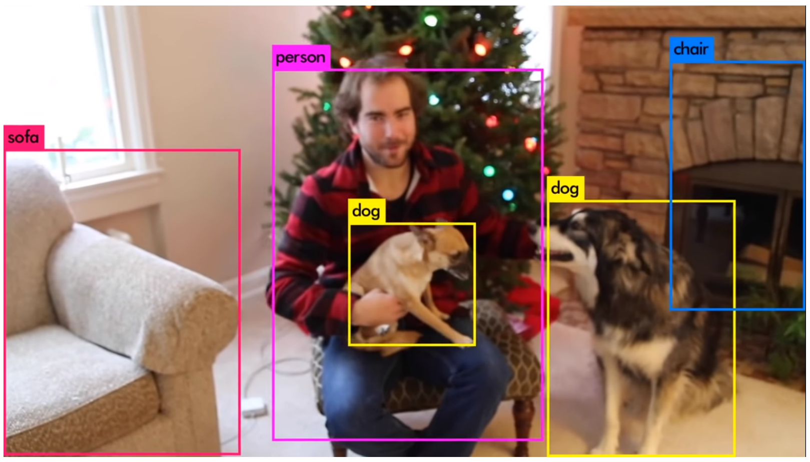


Figure 1.6: YOLO detection [3]

this sector. So it was quite challenging for us to work in this field.

1.3 Contribution

In our work, we explored the area of sign language detection and developed techniques to detect signs in real time by exploiting the *Faster Region-based Convolutional Neural Network* method (faster R-CNN) [9] and Long Short Term Memory(LSTM) model. Different sign languages are used worldwide, such as – American Sign Language (ASL), Argentinian Sign Language (LSA), Chinese Sign Language (CSL), Parisian Sign Language, etc. ASL recognition has been explored since around 1995 [10] and other languages are also investigated by different researchers ([11] [12]). However, very few works have been done on Bangladeshi Sign Languages (BdSL). Most of the previous works do not take advantage of CNN based object detection techniques to identify the gestures. In our thesis, we will investigate BdSL recognition and develop a system to help bridge the gap between signers and non-signers. Our work is mainly divided into two types. One is character based and the other one is word

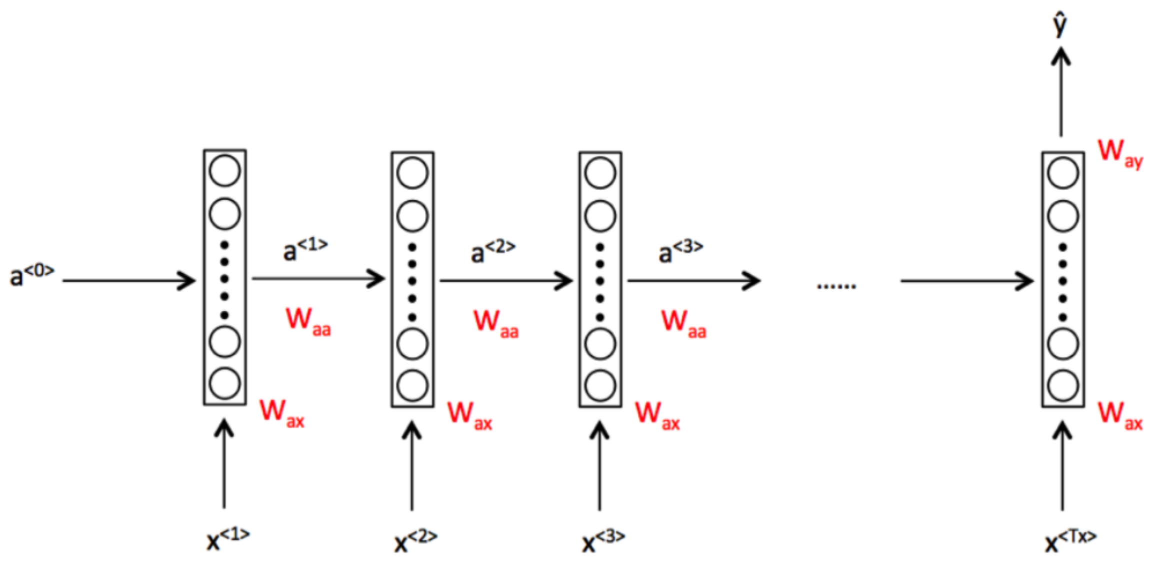


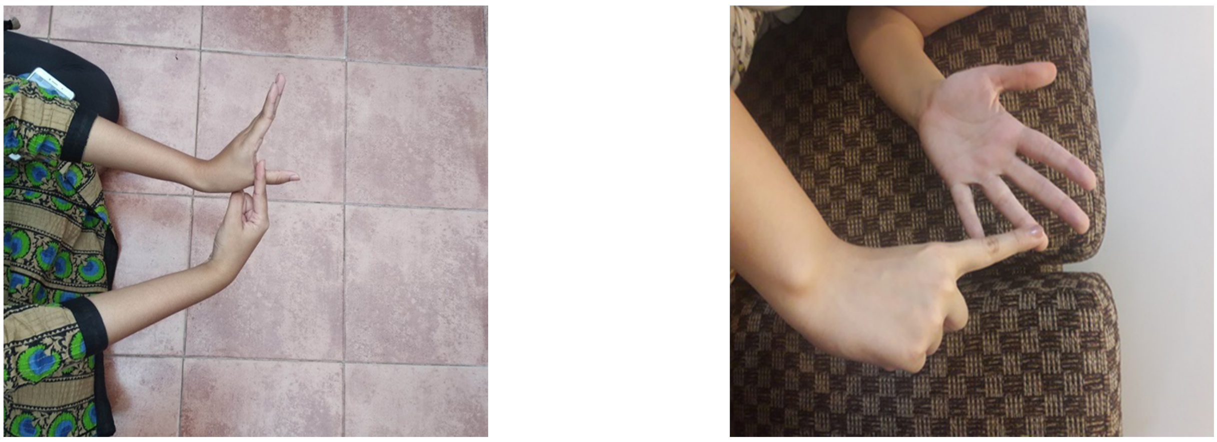
Figure 1.7: An one to many RNN model [4]. For video activity recognition, input(x1,x2,..) is video frames and output 'y' is the identification of the activity.

(action) based.
Developed countries have some work done in both of the fields but most of them require electric gloves or wires or sensors. On the contrary, we only need simple red colored gloves in our word recognition system and the sign letter based system is device independent. No real-time sign word analysis system using deep learning has been developed yet for BdSL and also most of the letter recognition systems are based on conventional machine learning methods. Apart from all the methods and architectures used, our main intention was reducing the communication gap between hearing impaired people and others. Our work was started with the detection of single letter recognition of Bangla Sign Language and we were successful in detecting the single letter from the real-time video and then we moved to word recognition. An overview of our contribution throughout our research is described next.

1.3.1 Sign Letter Recognition

As mentioned earlier, working on BdSL letter recognition in real time, a robust dataset is needed for deep learning methods. But there was no such dataset available. Some of our contributions in this research field are given below:

- We researched on BdSL sign language and other sign languages from several sources and visited 'Dhaka Badhir School' (located at Purana Paltan, Dhaka) where we verified our dataset and got consent for our research in this domain.
- We built a robust dataset and made it an open source for further research. Our dataset



(a) Showing sign ‘(ট)’.

(b) Showing sign ‘(এ)’.

Figure 1.8: Examples of *BdSLImset* dataset

BdSLImset specifics are described in Chapter 3. Two examples are given in Figure 2.14. The dataset is available here: https://github.com/annonymous3003/BdSLImset

- We implemented faster R-CNN model based deep learning approach in sign letter recognition.
- We built a real-time system of sign letter recognition with no background restriction. We have demonstrated the experimental outcomes of our proposed methodology.

1.3.2 Sign word Recognition

Only sign letters are not enough for communication. Real life application, however, does not work the way we perceived. BdSL is not just letters. Signers talk in gestures, that is, they make a series of hand movements to converse. For the testing purpose of our model, we went to "**Dhaka Badhir High School**" located at Purana Paltan, Dhaka. From there we got to know that it will be actually helpful to the signers if the system could be able to convert sign words (motion based recognition) instead of a single letter and this pushed us for improving our initial model and implementing the motion based Bangla Sign Language Detection. Our contributions to action based recognition are given below:

- We built a video dataset called ***BdSLVidset*** and made it open source for further research. Our dataset specifics are described in Chapter 3. A sample of our dataset is shown in Figure 1.9. The dataset is available here: https://github.com/Oishee30/BdSLVidSet.
- We implemented LSTM based deep learning approach in sign word recognition.

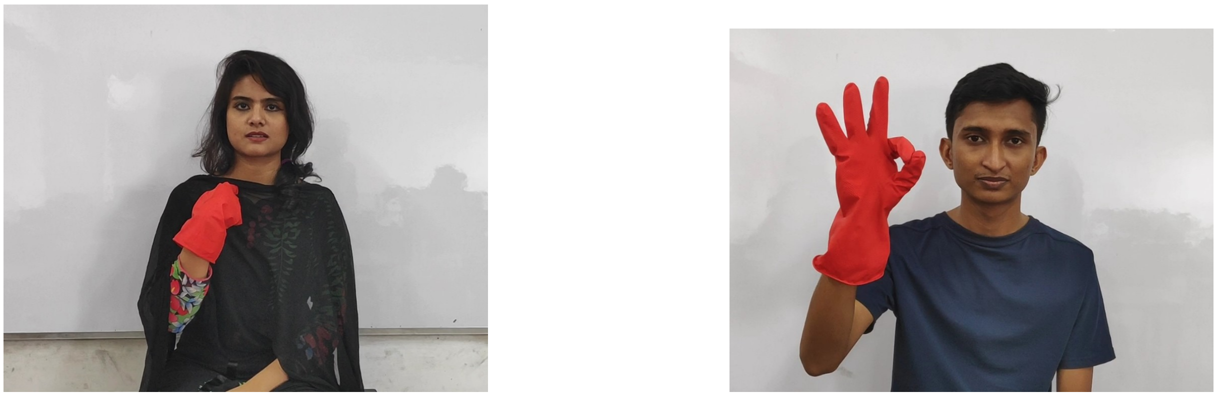

(a) Showing a frame of sign 'ami'. (b) Showing a frame of sign 'valo'.

Figure 1.9: Example frames from our video dataset.

- We built a real-time system of sign word recognition but there are some background and time limitations.

The initial research on real-time sign letter detection has been accepted in International Conference on Innovation in Engineering and Technology (ICIET) 27-29 December 2018, Dhaka, Bangladesh. Our paper, "**Real Time Bangladeshi Sign Language Detection using Faster R-CNN**" is available in **IEEE Xplore Digital Library** [13].

1.4 Sign Language Recognition Approach

We present a brief overview of our methodology in this section. Our sign letter recognition system - implemented using Faster R-CNN Inception V2 model - take frames as input from real-time videos. It searches for sign gestures, sub-dividing the frames in different regions. And if a sign gesture is found in the image, it displays a bounding box around it and predicts which sign is it. Another classifier, which classifies words is implemented using LSTM network model. It takes sequences of video frames as input. Then pre-process the frames and afterward each processed frame is fed into CNN for generating spatial feature values. Finally, these spatial feature values of each frame are then fed into an LSTM network as temporal features as input. This system then combines the features and predicts the word. Previous research works in detecting BdSL generally depend on external devices and most of the other vision-based techniques do not perform efficiently in real time. Our approach, however, is free from such limitations and the proposed methods are best suited to identify Bangladeshi signs in real time and recognize them successfully provided an enriched dataset is used.

1.5 Thesis Organization

The rest of the thesis advances as follows:

- Chapter 2 provides background and related works. Here we discuss the Deep Learning methods we studied and the papers which used such methods for developing systems that were helpful to us.
- The dataset preparation and processing is described fully chapter 3.
- In Chapter 4, how the CNN and RNN models extract information and predict output is described in detail.
- Our results and comparisons between different models and approaches used are described and explained in detail in chapter 5.
- The conclusion to our findings is given chapter 6 and further discusses all the possibilities this research can lead to.

1.6 Summary

BdSL language is an essential method of communication for hearing impaired people. So a system is needed for the effective communication between the normal people and the people who use sign language. There are two basic ways that BdSL is generally composed of, signs for letter recognition and gestures for word recognition. We have made dataset for both these sets and made it available for all. Also we have have a system for both the systems which recognizes letters and words rescpectively.

Chapter 2

Literature Review

In order to build robust classifiers for object and action detection, CNN based model and RNN based models are the two basic components of our systems. Each of the model architecture is reviewed below. Some similar existing systems to ours have also been reviewed with constructive comparisons.

2.1 Background Studies

In this section, we reviewed the necessary background needed to understand our proposed methodology for sign letters and word recognition systems. This consists of an architectural explanation of Neural Networks, CNN model, transfer learning concepts, Faster R-CNN architecture and LSTM model. We used Faster R-CNN in our research for letter recognition and LSTM for the detection of words.

2.1.1 Neural Network

Neural networks, as its name suggests, is a machine learning technique that is modeled after the brain structure [14]. It comprises of a network of learning units called neurons. These neurons learn how to convert input signals (e.g. picture of a sign letter) into corresponding output signals (e.g. the label 'A'), forming the basis of automated recognition. These neurons belong to the hidden layer. In a neural network that is trained with supervised learning, the training set contains values of the inputs as well as the target outputs labels. So the term hidden layer refers to the fact that in the training set, the true values for these nodes in the middle are not observed. The inputs and outputs are known but the things in the hidden layer are not seen in the training set (see Figure 2.1).

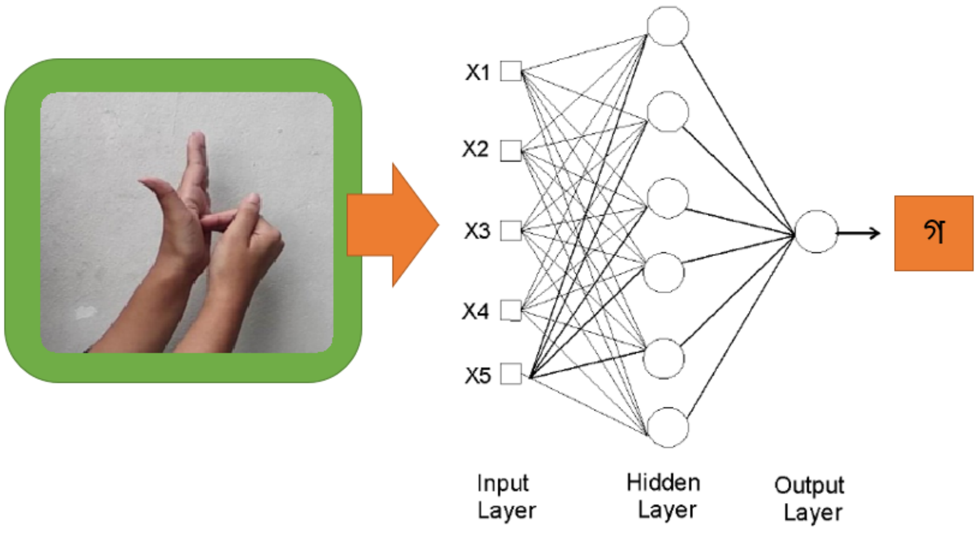


Figure 2.1: Suppose, we have the input features: x1, x2, x3, x4, x5 stacked up vertically from an image of sign letter. And this is called the input layer of the neural network. Then there's another layer of circles, called a hidden layer of the neural network. The final layer here is just one node. And this single-node layer is called the output layer and is responsible for generating the predicted value of the sign.

2.1.2 Convolutional Neural Network

One of the challenges of computer vision problems is that the inputs can get really big. For example, a 64 by 64 image consists 64 by 64 by 3 which is 12288 inputs because there are three color channels. And that's not too bad. But 64 by 64 is actually a very small image. Larger images may have 1000 pixel by 1000 pixel image, and that's just one megapixel. But the dimension of the input features will be 1000 by 1000 by 3, that's three million. For three million input features, the input layer will be three million dimensional. And so, if in the first hidden layers have just 1000 hidden units, then the total number of weights will be three billion parameters which are just very, very large. And with that many parameters, it's difficult to get enough data to prevent a neural network from over-fitting. And also, the computational requirements and the memory requirements to train a neural network with three billion parameters is just a bit unfeasible. To overcome this problem, convolution operations are needed, which is one of the fundamental building blocks of convolutional neural networks (see Figure 2.2). CNN consists of several of these Convolutional layers each of which can have a similar or different numbers of independent filters. All these filters are initialized randomly and become the model parameters that will be learned by the network subsequently.

Inputs from the convolution layer can be 'smoothed' to reduce the sensitivity of the filters

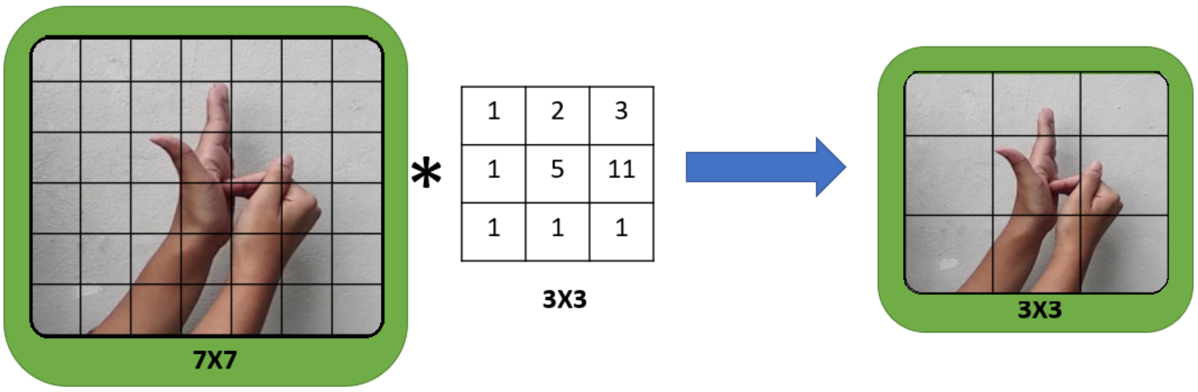


Figure 2.2: Example of the Convolution process in an image. A 7∗7 input image is convolved with a 3∗3 convolutional filter. This filter basically a feature extractor, which reduces the size of the image into a smaller one extracting the important features information.

to noise and variations. This smoothing process is called sub-sampling and can be achieved by taking averages or taking the maximum over a sample of the signal. Examples of sub-sampling methods (for image signals) include reducing the size of the image, or reducing the color contrast across red, green and blue (RGB) channels (see Figure 2.3).

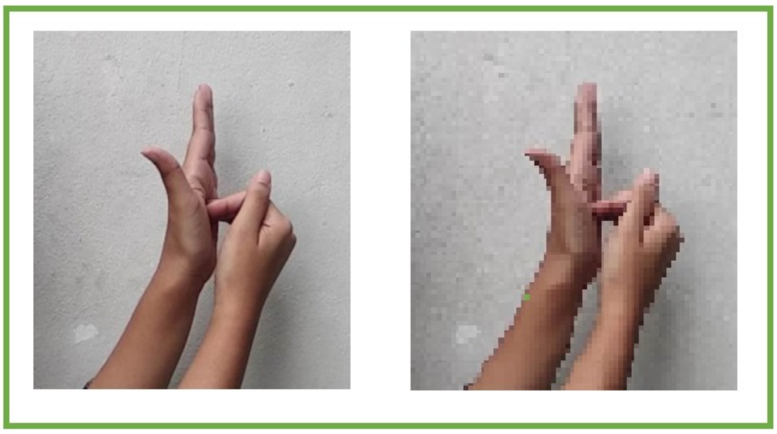


Figure 2.3: Examples of sub-sampling methods (for image signals) include reducing the size of the image or reducing the color contrast across red, green and blue (RGB) channels.

CNN also consists of pool layers. The most common approach used in pooling is max pooling in which the maximum of a region is taken as its representative (see Figure 2.4). The last layers in the network are fully connected neural network with improvised input values from convolutional layers, meaning that neurons of preceding layers are connected to every neuron in subsequent layers. This mimics high-level reasoning where all possible pathways

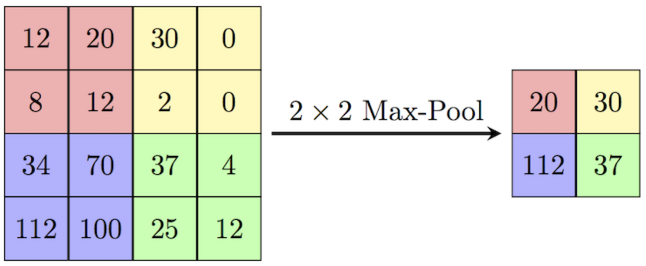


Figure 2.4: For example in the following diagram a 2x2 region is replaced by the maximum value in it [5].

from the input to output are considered. An example of overall CNN architecture is given in Figure 2.5.

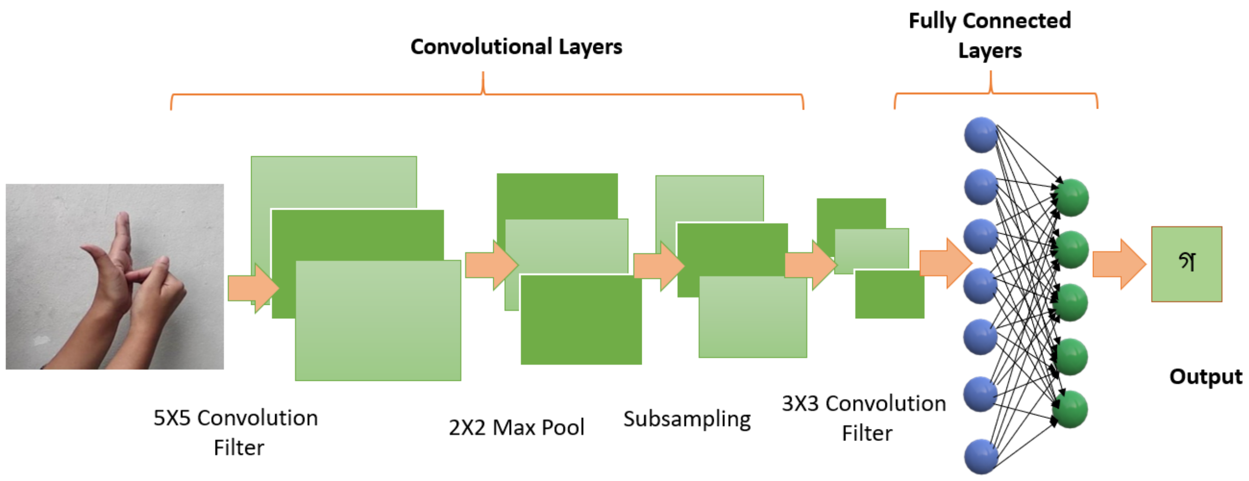


Figure 2.5: A CNN model consisting of different types of convolutional and fully connected layers.

Learning this different reasoning from images, the system requires feedback. This is done using a validation set (set of data not seen by the model yet) where the CNN would make predictions and compare them with the true labels or ground truth. The predictions in which errors are made are then fed back to the CNN to refine the weights learned, in a so-called backward pass. Formally, this algorithm is called back-propagation of errors, and it requires functions in the CNN to be differentiable.

2.1.3 Transfer Learning

In deep learning, transfer learning has become a really important part. As deep learning models are computationally expensive, these take a lot of time and processing power to start training from scratch. There are already a lot of pre-trained models, trained on huge datasets. These models consist of a huge number of hidden layers and pre-trained weights do not only work for the dataset they are trained on but also work as any type of feature detector. Such a model Faster R-CNN Inception V2, trained on the COCO dataset has been used in our system.

As mentioned earlier, sign letter recognition falls under the domain of image localization. That is, locating the sign in the image along with classifying it. Faster R-CNN is a region-based model that is best suited for this purpose. Now, a deeper elaboration of Faster R-CNN will be explained in the section below.

2.1.4 Faster R-CNN Architecture

The architecture of Faster R-CNN has several moving parts. First of all, we will discuss the high-level overview (shown in Figure 2.6) and then go over the details for each of the components.

It all starts with an image, from which we want to obtain:

- A list of bounding boxes.
- A label assigned to each bounding box.
- A probability for each label and bounding box.

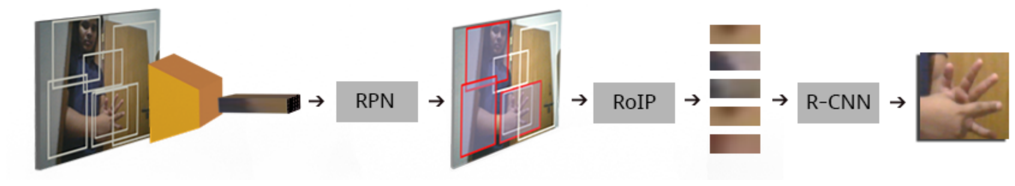


Figure 2.6: Complete Architecture of Faster R-CNN

The input images are represented as *Height*×*Width*×*Depth* tensors (multidimensional arrays), which has been passed through a pre-trained CNN until an intermediate layer, ending up with a convolutional feature map. We use this as a feature extractor for the next part.

Next, we have the Region Proposal Network (RPN). Using the features that the CNN computed, it is used to find up to a predefined number of regions (bounding boxes), which may contain objects.

The hardest issue with detecting an object is generating the variable-length list of bounding boxes. Usually, the last block is a fixed sized tensor output. In image classification, the output is a (N,) shaped tensor, with N being the number of classes, where each scalar in location i contains the probability of that image being label_i.

The variable-length problem is solved in the RPN by using anchors: fixed sized reference bounding boxes which are placed uniformly throughout the original image. Instead of having to detect where objects are, it divides the situation into two parts. For every anchor:

- Does this anchor contain a relevant object?
- How would it adjust this anchor to better fit the relevant object?

After having a list of possible relevant objects and their locations in the original image along with the features extracted by the CNN and the bounding boxes with relevant objects, we apply Region of Interest (**RoI**) Pooling and extract those features which would correspond to the relevant objects into a new tensor.

Finally, comes the R-CNN module which uses the above information to:

- Classify the content in the bounding box.
- Adjust the bounding box coordinates (so it better fits the object).

2.1.4.1 Base Network

As we mentioned earlier, the first step is using a CNN pretrained for the task of classification and using the output of an intermediate layer. Faster R-CNN used ZF and VGG pretrained on ImageNet but since then there have been lots of different networks with a varying number of weights. Here we talked about standard VGG-16 as an example.

Visual Geometry Group (VGG)
When using VGG for classification (Figure 2.7), the input is a 224×224×3 tensor (224×224 pixel RGB image). This has to remain fixed for classification because of the final block of the network uses fully-connected layers (instead of convolutional), which require a fixed length input.

Each convolutional layer creates abstractions based on previous information. The first layers usually learn edges, the second finds patterns in edges in order to activate for more complex shapes and so forth. Eventually, it ends up with a convolutional feature map (Figure 2.8) which has spatial dimensions much smaller than the original image, but greater depth. The width and height of the feature map decrease because of the pooling applied between convolutional layers and the depth increases based on the number of filters the convolutional layer learns. In its depth, the convolutional feature map has encoded all the information

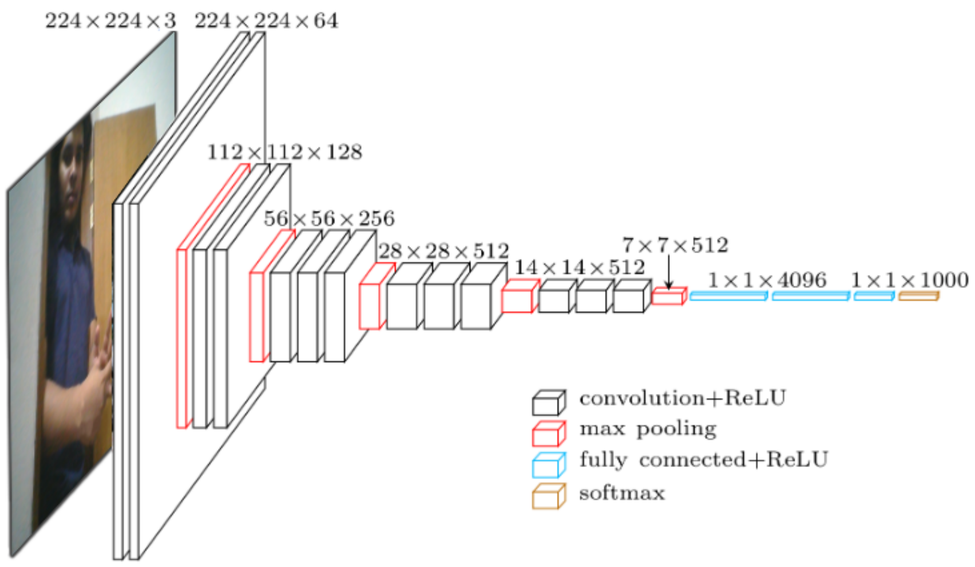


Figure 2.7: Visual Geometry Group classification process

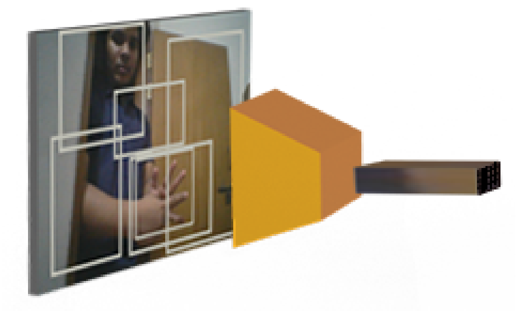


Figure 2.8: Image to convolutional feature map extraction

for the image while maintaining the location of the 'things' it has encoded relative to the original image. For example, if there was a red square on the top left of the image and the convolutional layers activate for it, then the information for that red square would still be on the top left of the convolutional feature map.

2.1.4.2 Anchors

Anchors are fixed bounding boxes that are placed throughout the image with different sizes and ratios that are going to be used for reference when first predicting object locations (Figure 2.9). Since we are working with a convolutional feature map of size $\text{Conv}_{\text{width}} \times \text{Conv}_{\text{height}} \times \text{Conv}_{\text{depth}}$, we create a set of anchors for each of the points in $\text{Conv}_{\text{width}} \times \text{Conv}_{\text{height}}$. Even though anchors are defined based on the convolutional feature map, the final anchors refer to the original image.

The dimensions of the feature map will be proportional to those of the original image. Mathematically, if the image was $w \times h$, the feature map will end up $w/r \times h/r$ where r is

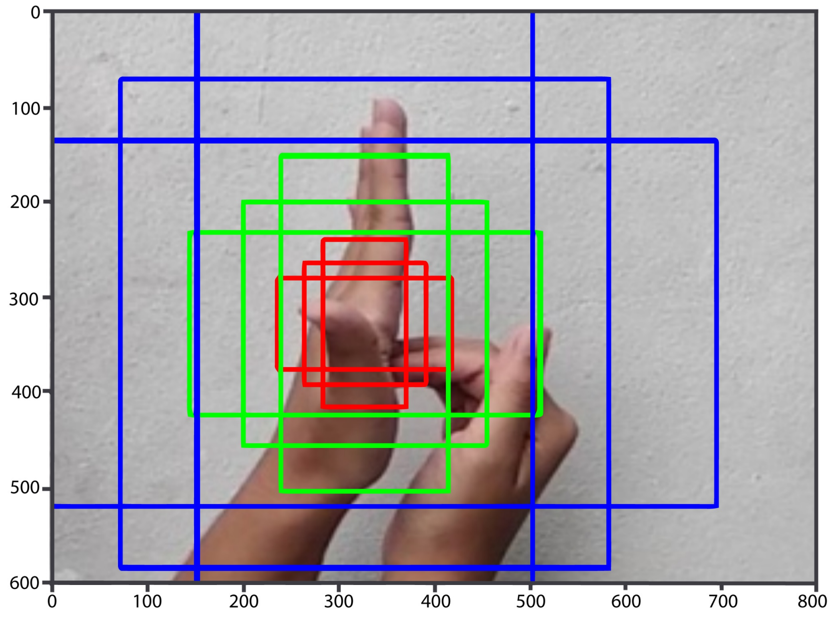


Figure 2.9: Anchor positions throughout Original Image

called sub-sampling ratio (weighted sum of the local region). If we define one anchor per spatial position of the feature map, the final image will end up with a bunch of anchors separated by *r* pixels. In the case of VGG, *r=16*. In order to choose the set of anchors we usually define a set of sizes and a set of ratios between the width and height of boxes and use all the possible combinations of sizes and ratios.

2.1.4.3 Region Proposal Network

RPN takes all the reference boxes (anchors) and outputs a set of good proposals for objects (Figure 2.10). It does this by having two different outputs for each of the anchors.

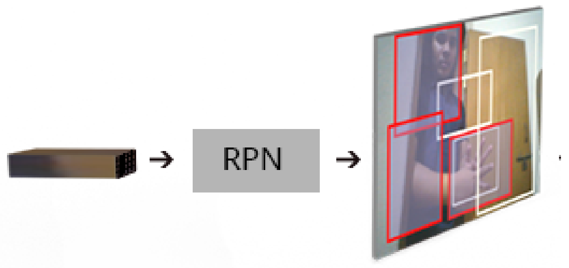


Figure 2.10: RPN uses feature map and generates proposal over image

The first one is the probability that an anchor is an object. RPN does not look for what class of object it is, only that it does, in fact, look like an object (and not a background). In another way, we can say it looks for *'objectness score'*. This objectness score filters out the bad predictions for the second stage.

The second output is the bounding box regression for adjusting the anchors to better fit the object it is predicting. Using the final proposal coordinates and their *'objectness score'* it brings out a good set of proposals for objects.

2.1.4.4 Region of Interest Pooling

After the RPN step, we have a bunch of object proposals with no class assigned to them. Next, we have to take these bounding box and classify them into desired categories. The simplest approach would be to take each proposal, crop it, and then pass it through the pre-trained base network. The main problem is that running the computations for a huge number of proposals with different sizes.

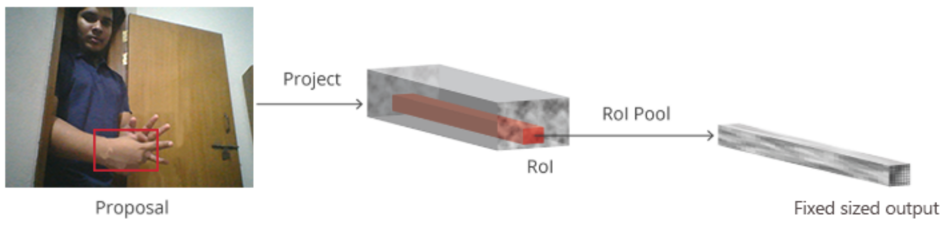


Figure 2.11: Region Of Interest Pooling process

Region of Interest Pooling can simplify the problem by reducing the feature maps into the same size. Unlike Max-Pooling which has a fixed size, RoI Pooling splits the input feature map into a fixed number (Figure 2.11) of roughly equal regions, and then apply Max-Pooling on every region. Therefore the output of RoI Pooling is always a fixed number regardless of the size of the input. With the fixed RoI Pooling outputs as inputs, we have lots of choices for the final architecture level (*R-CNN*).

2.1.4.5 Region-based Convolutional Neural Network

Region-based convolutional neural network (*R-CNN*) is the final step in Faster R-CNN's pipeline.

After getting a convolutional feature map from the image, using it to get object proposals with the RPN and finally extracting features for each of those proposals (via RoI Pooling), we finally need to use these features for classification. The *R-CNN* takes the feature map for each proposal, flattens it and uses two fully-connected layers. Then, it uses two different fully-connected layers (Figure 2.12). For each of the different objects R-CNN has two different goals:

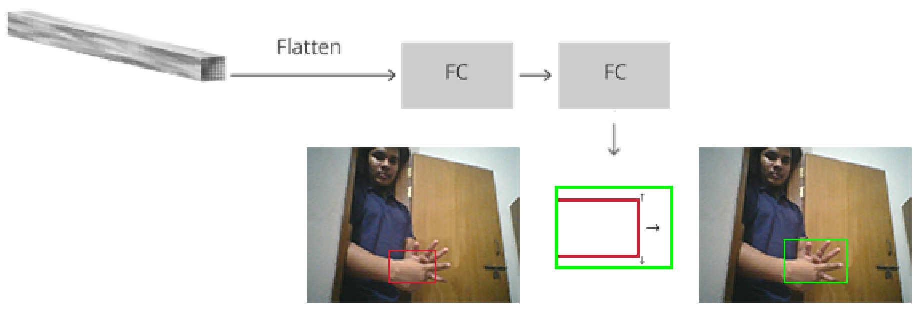


Figure 2.12: R-CNN, shaping the final output

- Classify proposals into one of the classes, plus a background class (for removing bad proposals).
- Better adjust the bounding box for the proposal according to the predicted class.

2.1.5 Long Short-Term Memory Architecture(LSTM)

LSTM is a recurrent neural network (RNN) architecture that can remember the values over arbitrary intervals. Where there is a time series of unknown duration, LSTM is well-suited to classify, process and predict. A chain of repeating modules of the neural network is followed by all current recurrent neural networks. In standard RNNs, the repeating modules have a structure with a single layer. LSTM also has a chain structure where the repeating module has a different structure. Instead of having a single neural network layer, there are four.

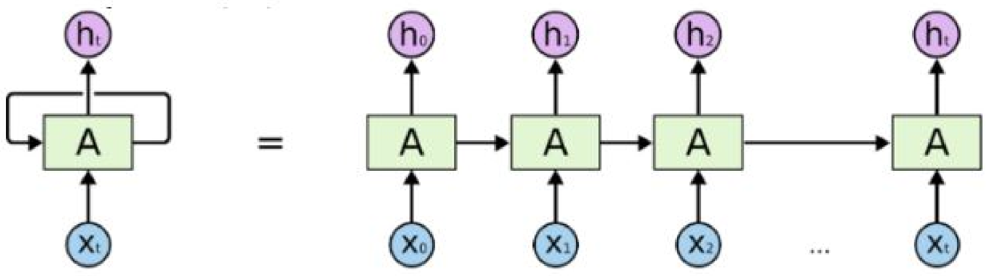


Figure 2.13: LSTM Architecture [6]

The structure of the RNN and the hidden Markov model is very similar. The main difference is calculating and constructing parameters. Insensitivity to gap length is an advantage of LSTM. RNN and HMM rely on the hidden state before emission/sequence. If we want to predict the sequence after 1,000 intervals instead of 10, these models forget the starting point but LSTM remembers [4].

2.1.5.1 Architecture which allows LSTM to Remember

RNN cell takes in two inputs, output from the last hidden state and the observation of current time. Apart from the hidden state, there is no information about the past which is remembered.

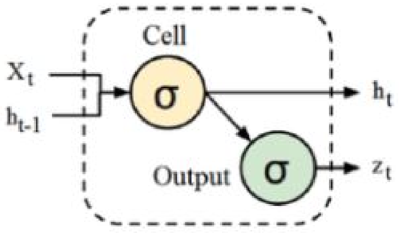


Figure 2.14: RNN Cell [7]

The long-term memory is also called by cell state. The looping arrows indicate the recursive nature of the cell. It permits to store the previous info found from intervals. The cell state is modified by the forget gate placed below the cell state and also adjusted by the input modulation gate. With the use of the equation, the previous cell state forgets by multiple with the forget gate and adds new information through the output of the input gates.

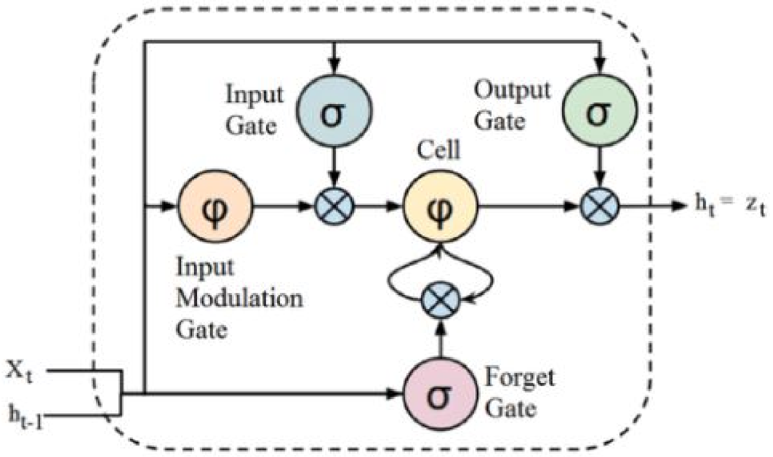


Figure 2.15: LSTM Cell [7]

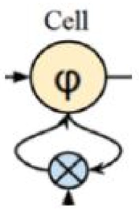


Figure 2.16: Cell State [7]

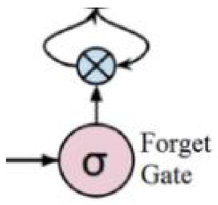


Figure 2.17: Forget Gate [7]

The remember vector is usually called the forget gate. The output of the forget gate tells the cell state which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell state. In weighted input/observation and previous hidden state, the sigmoid function is applied.

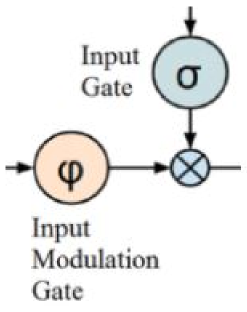


Figure 2.18: Input Gate [7]

The save vector works as an input gate, determines which information should enter the cell state / long-term memory. An activation function is required for each gate. The input gate is a sigmoid function and has a range of [0,1]. Because the equation of the cell state is a summation between the previous cell state, sigmoid function alone will only add memory and not be able to remove/forget the memory.

The number will never be zero / turned-off / forget if we can add a float number between [0,1]. The input modulation gate has a tanh activation function with a range of [-1, 1] which allows the cell state to forget the memory.

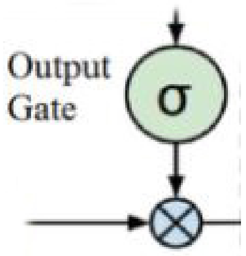


Figure 2.19: Output Gate [7]

Another name of the output gate is a focus vector. From all the possible values of the matrix, which should be moving forward to the next hidden state and what information should be passed to the next sequence.

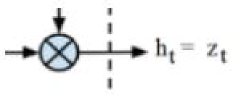


Figure 2.20: Hidden State [7]

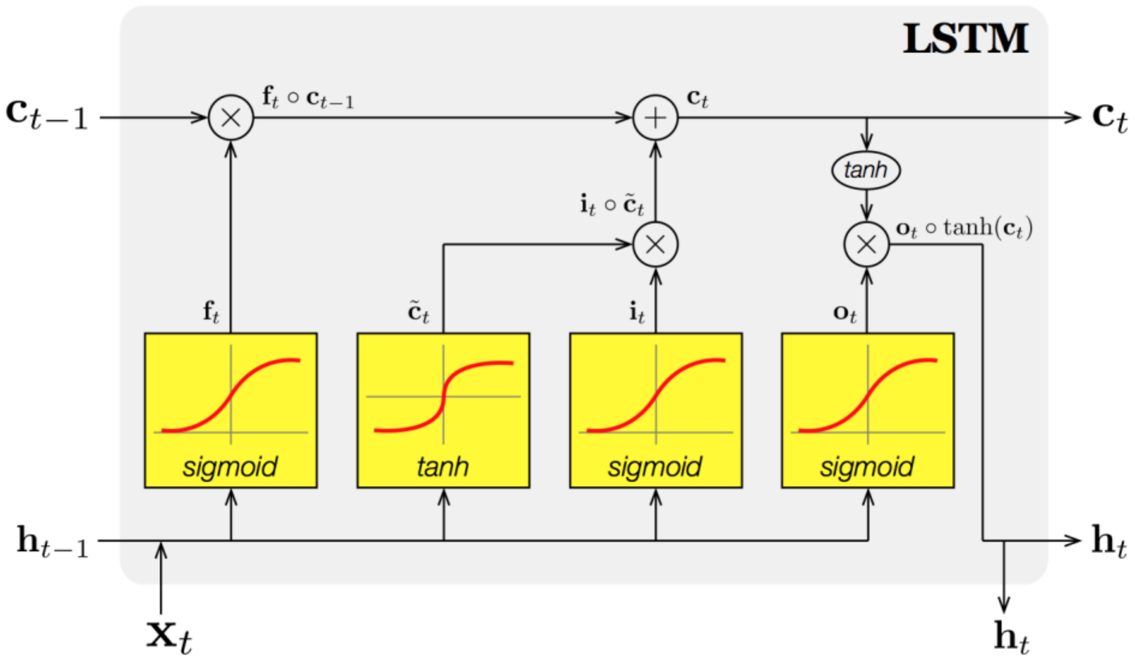


Figure 2.21: LSTM Cell with activation [6]

The first sigmoid activation function is the forget gate. It identifies which information should be forgotten from the previous cell state (Ct-1). The input gate includes a second sigmoid and first tanh activation function. The remaining last sigmoid considered as an output gate that highlights the information going to the next hidden state.

2.2 Related Works

In this section, we will discuss the works done on Bangladeshi Sign Language recognition so far and the works that are related to our work. Discussed below is an overview of the papers, that we have gone through, with their process, limitations, and challenges the authors faced during their experiment. We have reviewed the papers in a separate section, one of them is related to sign letter recognition and the other one is related to action based recognition.

2.2.1 Object Recognition

In this section, some papers related to object and sign letter recognition have been reviewed.

2.2.1.1 Bangla Sign Language Recognition using Convolutional Neural Network (2017)

Topic:

This paper proposed a system that uses a learning-based approach to recognize BdSL. They used **Leap Motion Controller** and **CNN** for their work [15].

Project Details:

The leap motion controller was used to track the continuous motion of the hands. It provides a skeletal model of the hand with appropriate data of hand position, orientation, rotation, fingertips, grabbing and more non-linear features. It preprocessed all the data by removing garbage collection and creating virtual hands in virtual reality. A set of layer was used to process the data obtained from the leap motion controller. Basically, Convolutional Neural Network was used here to build a decision network. How it worked layer-wise is shown in Figure 5.1. The features obtained from the leap motion controller are sent as a parameter

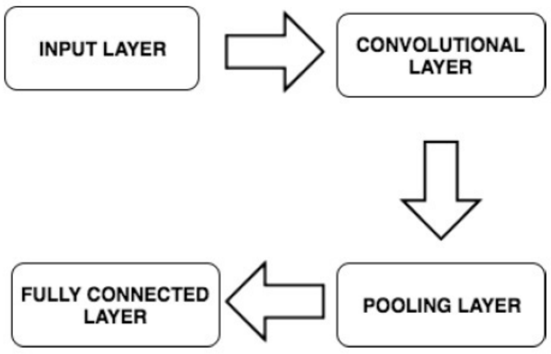


Figure 2.22: Layers on processing BdSL features. *(Source: Original Paper)*

to the next layer which is the convolutional layer. The finger features are considered as a hyper-parameter. After the Convolutional layer, they passed the output to the pooling layer which measured the comparison and the loss function and forwarded them to the fully connected layer.

Limitations:

Though they have only a 3% error rate, the main problem is they used an external device for their work. This reduces the user-friendliness of the system and the approach is not a real-time approach.

2.2.1.2 A Real-Time Appearance-Based Bengali Alphabet And Numeral Signs Recognition (2017)

Topic:

This paper presented a system that recognizes the Bengali alphabet and numeral signs. They used region of interest and **K-nearest Neighbours (KNN)** classifier for the work. [16]

Project Details:

They initialized Region of Interest (ROI) by detecting 'Opened Hand' followed by 'Closed Hand' posture from captured images. The area was segmented based on the Hue and Saturation values of human skin. Then they were converted to binary images followed by row vectors.

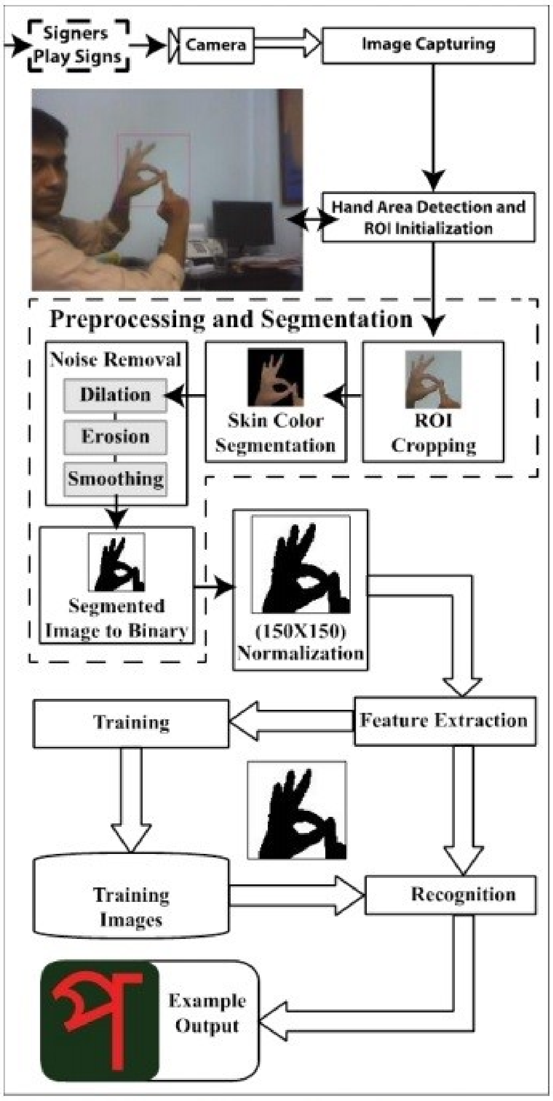


Figure 2.23: Block diagram of the proposed system. *(Source: Original Paper)*

The KNN classifier was used for training. It was used for storing featured vectors of normalized binary images and class labels of training data sets. Separate KNN classifiers were trained for the Bengali alphabet and numeral signs with different K values. The architecture is shown in Figure 2.23. For their dataset, they used 3600 images of 36 Bangla alphabets and 1000 images of numeral signs. 6 vowels and 30 consonants were used. Their images were performed by 10 people, 6 are male and 4 are female. Accuracy for BdSL alphabets was 91.73% and for numeral recognition, it was 96.53%.

Limitations:

They faced a problem in differentiating 'র' and 'ল'. They got only 72-73% accuracy for these letters. All their images were on the same background which is a downside as it reduces versatility.

2.2.1.3 Bangladeshi Sign Language Recognition Employing Neural Network Ensemble (2012)

Topic:

This system proposed a Bangladeshi Sign Language recognizer using **NCL** algorithm. It can interpret Bangladeshi Sign Language into Bengali text and vice-versa [17].

Project Details:

The process started with the detection of the skin color of hand in front of a webcam to capture the images of BdSL. It was done for different environments. The images were converted into threshold value and then normalized to 30 × 33 scale pixels. Then they applied the feature extraction method and they applied NCL algorithm to train those images. They have an average accuracy of 93%. The whole block diagram is shown in Figure 2.24.

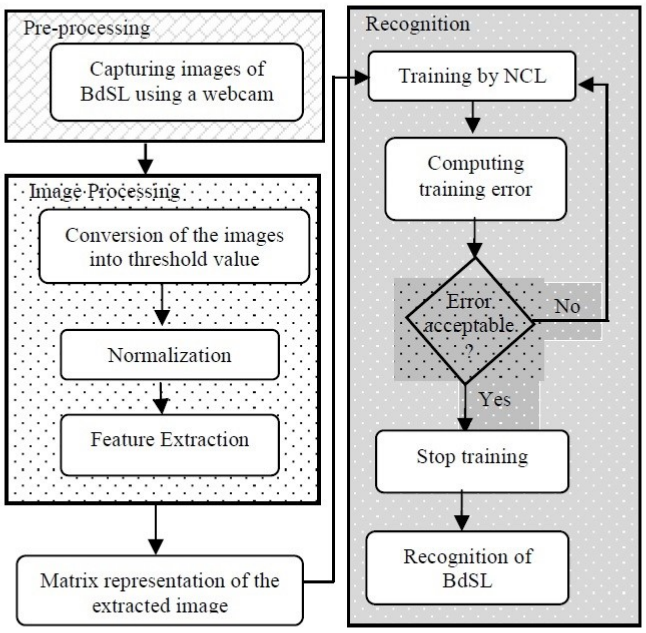


Figure 2.24: Block diagram of the proposed BdSLR. *(Source: Original Paper)*

Limitations:

This method of recognition needs conversion, pre-processing and feature extraction of image dataset.

2.2.1.4 An Automated Bengali Sign Language Recognition System Based on Fingertip Finder Algorithm (2015)

Topic:

This paper presents a new algorithm to identify Bengali Sign Language (BdSL) for recognizing 46 hand gestures. They proposed a system that will recognize alphabets and numerals of BdSL using the [16] **fingertip finder algorithm**. **Project Details:**

At first, they constructed a database of 2300 images of Bengali signs. Images were resized into 260×260 pixels and converted to binary. Otsu's thresholding method was used for this conversion. After cropping the region of interest then the feature extraction was started. Feature extraction stage was implemented by finding the points of visible fingertips and the centroid of the hand region. Based on these points, ten features were extracted from each image. Then they applied the fingertip finder algorithm to find the positions of the fingertips. The architecture is shown in Figure 2.25.

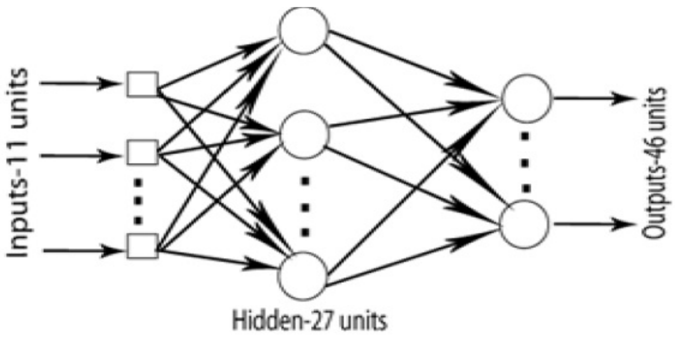


Figure 2.25: ANN architecture *(Source: Original Paper)*

Multilayered feed-forward neural networks were used with a back-propagation training algorithm to identify the Bengali sign gesture. Each sign of the alphabet was represented by a vector containing 11 features. The input layer had 11 unit inputs and the output layer was made of 46 units, as BdSL sign language has 46 static signs. Their average recognition rate is 88.69%.

Limitations: Not a real-time approach and recognition rate is not high.

2.2.1.5 Bangladeshi Sign Language Recognition using Fingertip Position (2016)

Topic:

They investigated a BSL recognition scheme based on the fingertip position. The method considered relative tip positions of five fingers in two dimension space and position vectors are used to train **Artificial Neural Network (ANN) for recognition purpose** [18].

Project Details:

At first, they developed a dataset of real hand gesture and then proposed BSL, based on fingertip position (BSL-FTP), is applied to it. The basic steps of BSL-FTP are pre-processing of gesture images, generation of tip positions data and finally training ANN with tip position of fingers. As relative positions of individual fingers are different in different letters, tip position values are useful to recognize a letter. ANN is used to train with position values for recognition purposes. As shown in Figure 2.26, three-layered ANN architecture is con-

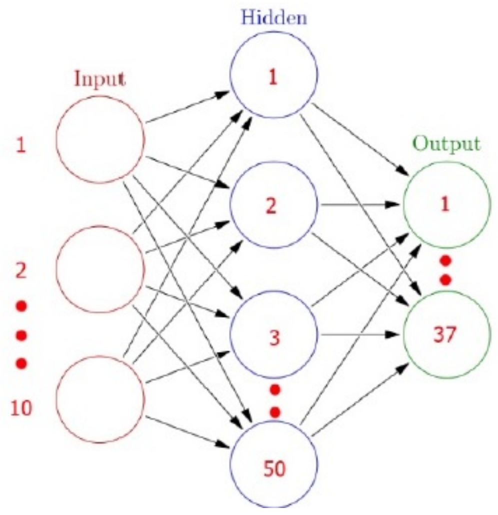


Figure 2.26: ANN architecture for BSL-FTP *(Source: Original Paper)*

sidered in their study. Around 60% of total data is used for training ANN and the rest was reserved to measure the performance. Over a vocabulary of 40 ASL words, they gained an accuracy of 95.4%.

Limitations:

This proposed methodology doesn't work in real-time.

2.2.2 Action Recognition

In this section, some papers related to action and sign word recognition have been reviewed.

2.2.2.1 Sign Language Learning System with Image Sampling and Convolutional Neural Network (2017)

Topic:

This paper worked with consecutive movements of a hand within a time period and gave them a single meaning using **CNN** [19].

Project Details:

At first, a video of sign language demonstration is sampled and concatenated into an image. Data is created directly from the demonstration video as it is easy to add recognizable motion. After that, the image became the input of the convolutional neural network (CNN). From the training data, CNN selects the candidate with high probabilities. Their flow diagram is shown in Figure 2.27.

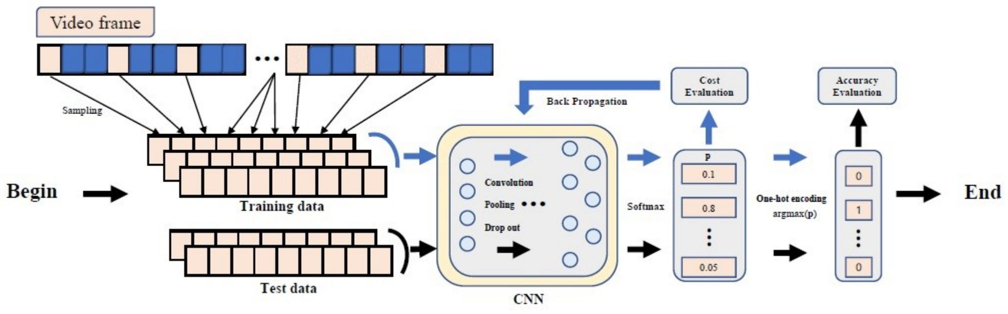


Figure 2.27: Flow diagram of the system. *(Source: Original Paper)*

They selected six various sign language actions for the experiment. They had a total of 21 data sets among them 20 were used as learning data and one was test data.

The main advantage of this method is it uses only 2D images so it can easily be done with a cheap camera. The size of the training data is relatively small because the images are sampled.

Limitations:

But the accuracy is relatively low, compared to other techniques it is only 86%.

2.2.2.2 Real-time American Sign Language Recognition with Convolutional Neural Networks (2016)

Topic:

This paper presented the development and implementation of an American Sign Language (ASL) finger-spelling translator based on **CNN** [10].

Project Details:

Their system translates the video of ASL signs of a user to text. For their input, they obtained videos of the user signing. Then classified each frame in the video to a letter. The classification is done using a convolutional neural network (CNN). They used a soft-max based classification. Their dataset is divided into two types: color images and depth images. The dataset of color images is made of 24 static signs of ASL captured by 5 users and it contained over 65,000 images. They used 2,524 close up images for depth images that were placed over a uniform black background as shown in Figure 2.28.

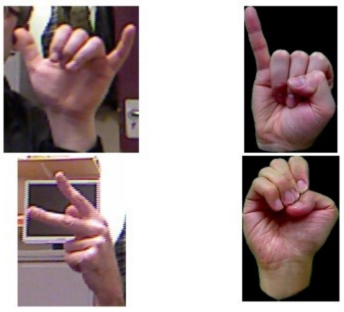


Figure 2.28: Dataset examples. Left: Surrey University. Right: Massey University. Top left: ‘y’. Bottom left: ‘k’. Top right: ‘i’. Bottom right: ‘e’. *(Source: Original Paper)*

Limitations:

Their dataset was on the same background. They were failed to train the letter ‘j’. Their accuracy suffers for the letters ‘k’ and ‘d’.

2.2.2.3 Real-Time Sign Language Gesture (Word) Recognition from video sequences using CNN and RNN (2018)

Topic:

They have made an application by which sign language gestures can be recognized. They have trained the model on spatial features and used the inception model which is a deep convolutional neural network (**CNN**). Furthermore, they have used the recurrent neural network (**RNN**) to train the model on temporal features [8].

Project Details:

Sign language is a sequence of gestures that generate a meaningful sentence. They worked with multiple gestures of Argentinean Sign Language. They extracted the keyframe, based on gradient, to split the video to independent isolated gesture. The features were extracted from gestures by applying Orientation Histogram and Principal Component Analysis. Correlation, Manhattan, and Euclidean distance were used for classification. They found better accuracy using correlation and Euclidean distances. CNN and RNN have been used in their work.

CNN has been used for capturing local spatial patterns in the data. They are great at finding patterns and then use those to classify images.

RNN is used for recognition tasks. They have used Long Short-Term Memory (LSTM) which is a variation of RNN with LSTM units. The first layer feeds input to the upcoming layer. Their model is a wide network consisting of a single layer of 256 LSTM units. This layer is followed by a fully connected layer with soft-max activation. Finally, a regression layer is applied to perform a regression to the provided input. The detailed flow of layers is given in Figure 2.29.

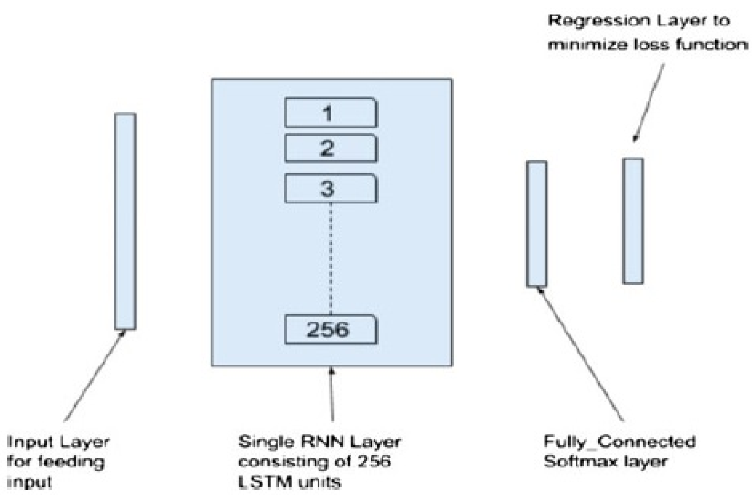

Regression Layer to minimize loss function
1
2
3
256
Input Layer for feeding input
Single RNN Layer consisting of 256 LSTM units
Fully_Connected Softmax layer

Figure 2.29: Full flow of layer-based work. *(Source: Original Paper)*

They have used the prediction approach and pool layer approach for their method.

In the prediction approach, spatial features for individual frames were extracted using the inception model (CNN) and temporal features using RNN. In the pool layer approach, they used CNN to train the model on the spatial features and passed the pool layer output to the RNN before it is made into a prediction.

They obtained an accuracy of 95.217% and showed that CNN along with RNN can be successfully used to learn spatial and temporal features and classify sign language.

Limitations:

Using a high number of features has improved the correct classification rate for most of the gestures, but that causes the RNN to discover random noise in the finite training set. Hence, it learned some features that did not add any value to the judgment and have led to overfitting. The performance using the prediction approach is not up to the mark.

2.2.2.4 An Unsupervised Approach for Human Activity Detection and Recognition (2013)

Topic:

In this paper, they used K-means clustering and simple template models to achieve human activity detection and recognition in an unsupervised manner. They extracted the features from the skeleton data obtained from an inexpensive RGBD (RGB-Depth) sensor [10].

Project Details:

They have worked with human activity recognition and they have used unlabeled observations for their purpose. They have used the data from an RGBD sensor to deal with unlabeled observations. Their work is based on visual data. With unsupervised human activity recognition, an intelligent system can autonomously detect new activities. This allows the system to function autonomously without introducing new activity models which are the case of supervised learning.

Their work is done on three steps. Extracting features from skeleton data, activity detection by clustering and then activity detection using templates.

For extracting features from skeleton data at first they obtained the raw data which coordinates directly from the RGBD sensor. In their work, each activity observation is sampled for a window of two seconds. At 30fps, each activity observation contains 60 frames at full resolution. The features are formed based on a human's range of movements. For each pose in each frame, the features are extracted from the coordinates of the different joint positions. The vectors are formed locally (between joints) and normalized, scale invariant to the size of the subject.

Activity detection finds new activities. They used K-means clustering to group similar activities and discriminate one from the other.

To model each activity they used the centroid of the cluster as the template for each activity.

Limitations:

The first is the ability of the system to determine the number of clusters, (k value) by itself. There are a number of ways to determine the k value including the use of cluster validity indices.

Another issue is the confusion of highly similar activities. Their results show an average detection performance of 80.4% precision and 83.8% recall.

2.2.2.5 Unsupervised Learning for Physical Interaction through Video Prediction (2016)

Topic:

In this paper, they developed an action-conditioned video prediction model that explicitly models pixel motion, by predicting a distribution over pixel motion from previous frames. Their model explicitly predicts motion and it is partially invariant to object appearance enabling it to generalize to previously unseen objects [10].

Project Details:

Learning to predict physical phenomena poses many challenges since real-world physical interactions tend to be complex. They proposed a method which does not require the model to store the object and background appearance. Such appearance information is directly available in the previous frame. They developed a predictive model that merges appearance information from previous frames with motion predicted by the model. As a result, their model is better able to predict future video sequences for multiple steps, even involving objects not seen at training time. They have used Dynamic Neural Advection (DNA), Convolutional Dynamic Neural Advection (CDNA) and Spatial Transformer Predictors (STP) for their work.

With DNA they predicted a distribution over locations in the previous frame for each pixel in the new frame and the predicted pixel value is computed as an expectation under this distribution. They kept the dimensionality of the prediction low by doing this.

With CDNA they predicted the motions of different objects in different regions of the image. Instead of predicting a different distribution for each pixel, their model predicts multiple discrete distributions that are each applied to the entire image via a convolution, which computes the expected value of the motion distribution for every pixel. Their work is based on the idea that pixels on the same rigid object will move together, and therefore can share the same transformation.

With STP their model produced multiple sets of parameters for 2D affine image transformations and applied the transformations using a bi-linear sampling kernel.

Limitations:

Their model directly predicts the motion of image pixels and naturally groups together pixels that belong to the same object and move together but it does not explicitly extract an internal object-centric representation.

2.3 Summary

In this chapter, firstly we have have described the necessary architectures to demonstrate our system in further chapters. These architectures include Neural Network, Convolutional Neural network, Transfer learning, Faster R-CNN, Long Short Term Memory(LSTM). Then we reviewed a total number of 10 research articles and discussed about their working procedures. These reviews are subdivided into two other sections - one section contains objection recognition related reviews and another section is related to action recognition based reviews. We have summarized their work and also mentioned their limitations.

Chapter 3

Our Dataset

Our project involves the recognition of Bangladeshi Sign Language (BdSL) using deep learning methods. While investigating this research domain, we found a lacking of a proper BdSL dataset to integrate with deep learning models. So, we had to build our own dataset. We have divided our datasets into two groups - an image dataset for sign letter recognition, and another is a video dataset for sign word recognition. Our datasets are designed focusing on the necessity of training deep learning models discussed in the earlier section.

3.1 BdSLImset

BdSL letters are difficult, consist of different gestures. Also, some of the letters have similar gestures, hard to differentiate for a computer vision system. Large datasets can give a higher accuracy and confidence rate in recognition of sign letters. Some examples of BdSL letters has been shown in Figure 3.1.

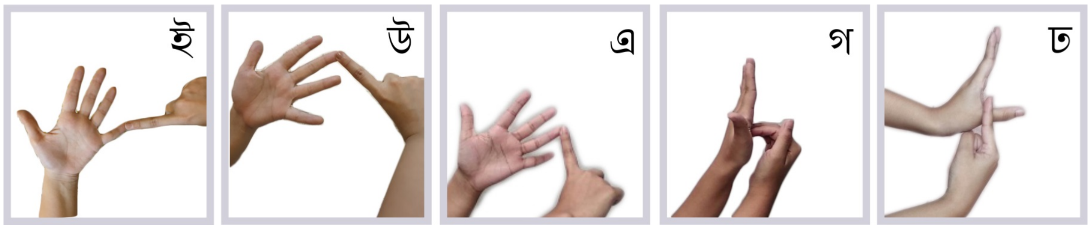


Figure 3.1: Different signs of BdSL letters are shown here.

Real-time Sign Letter recognition can be fit into the domain of image localization. The system has to find if a sign exists in the image and if it exists which sign it is. For this purpose the training datasets must consist of :

- Robust Background - with various lighting conditions and dynamicity.

- Gestures must have variation in angles from multiple subjects.
- Bounding box variables (Xmax, Xmin, Ymax, Ymin) to detect where the sign gesture is in the image. An example of a bounding box on the image is shown in Figure 3.2.

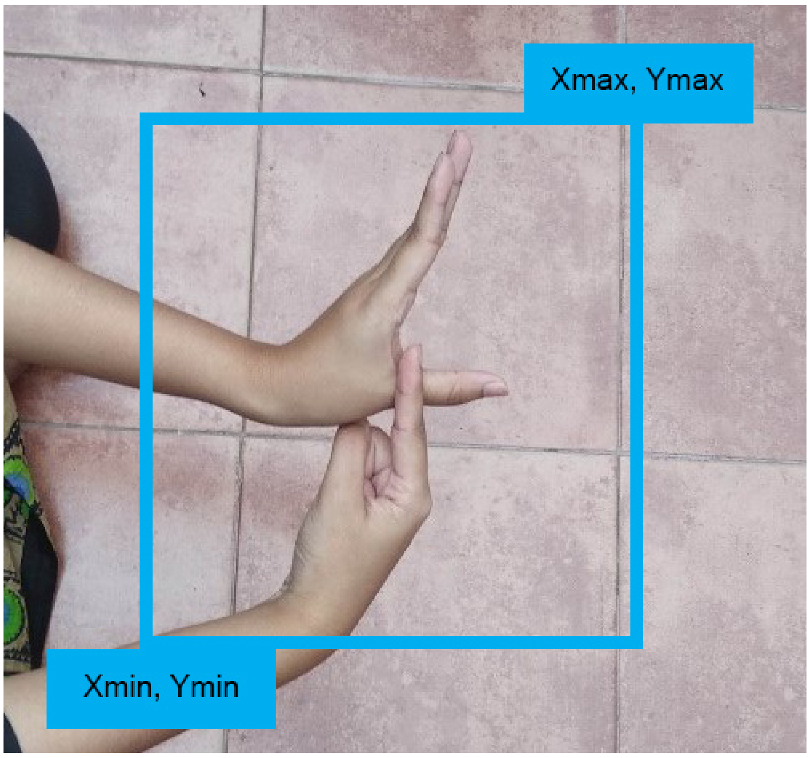


Figure 3.2: Bounding box detecting a sign.

3.1.1 Bangladeshi Sign Language Image Dataset - *BdSLImset*

As we're working on BdSL recognition in real time, there can be thousands of different backgrounds and variations in recognition time. For this purpose, the dataset had to be very enriched to build such a robust classifier. Training images needed to have variations in the signs of letters, the backgrounds and also the lighting conditions. Therefore, we kept these factors in considerations while collecting images for this dataset. We created a dataset- *BdSLImset*. Figure 3.3 shows some sample images of *BdSLImset*.

- The gestures are recorded in multiple orientations, i.e. in different angles to train our model to recognize the letters in any variation.
- As we know that our testing environment in real-time can have many backgrounds, to recognize them correctly our dataset has multiple backgrounds and lighting scenarios.
- The images were prepared with people of different ages, gender, etc.

Figure 3.4 shows variations of each class in the *BdSLImset*.

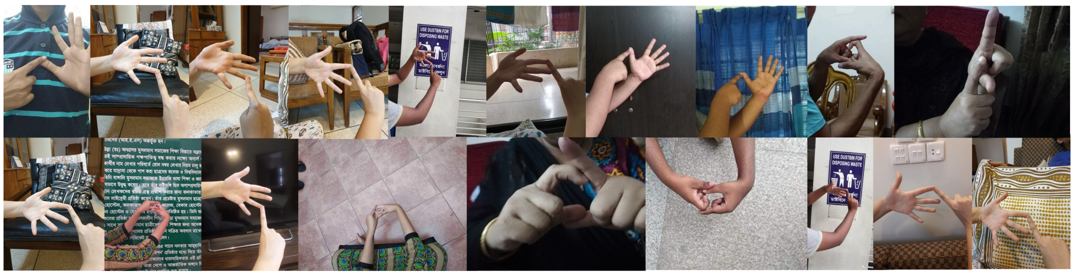


Figure 3.3: Sample sign language images in *BdSLImset* dataset for different sign letters with different background and other variations.

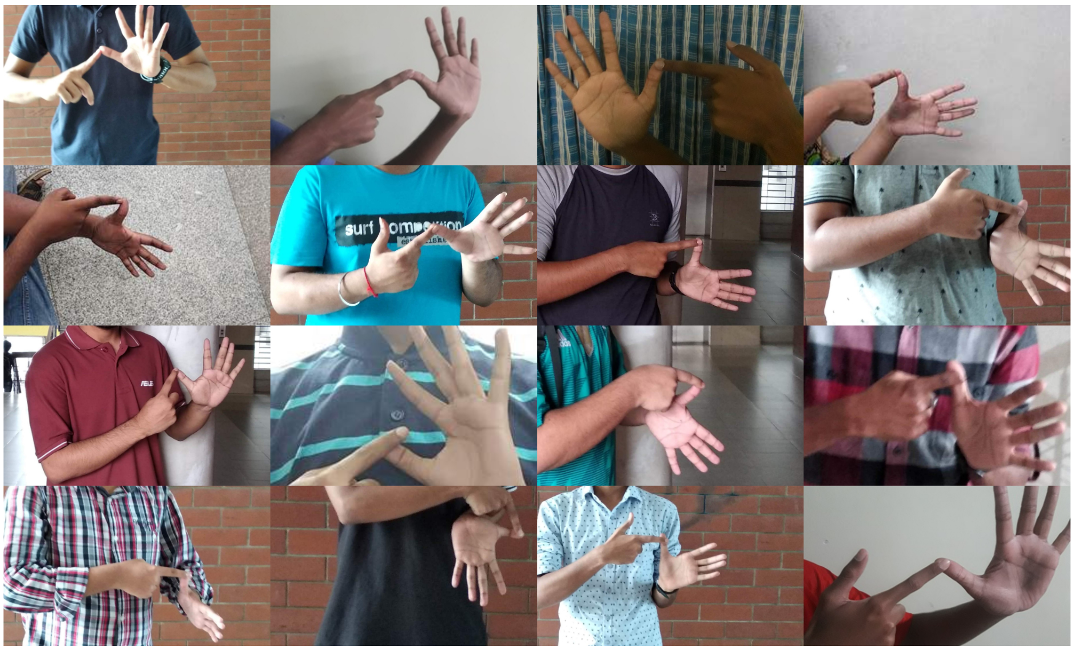


Figure 3.4: Sample sign language images in *BdSLImset* with dynamic background. These images belong to class 'আ'.

3.1.2 Dataset Specifics

Each image size is less than 200kb and the resolution is not more than 700×1280. Currently, our *BdSLImset* dataset has 10 different labeled sign letters. We collected about 200 pictures for each gesture. For each letter about 200 sign images of 20 persons of different ages and genders have been captured with varieties of backgrounds. The dataset is divided into a training set and testing set with a ratio of 8:2. After gathering images, we selected the region of each of the hand gestures with a bounding box and labeled them (see Figure 3.5). Thus the initial training data are prepared.

All subjects were right-handed non-signers and were taught how to perform the signs during the recording session by showing them a video of the signs as well as performed by one of the authors. All subjects practiced each sign a few times before recording.

Our dataset is verified by the teachers of **Dhaka Bodhir School**. Academic, educational or personal use of our dataset is allowed without restrictions.

An overall review of BdSLImset has been given in Table 3.4 and the letters with corresponding 'ID' are shown in Table 3.5.

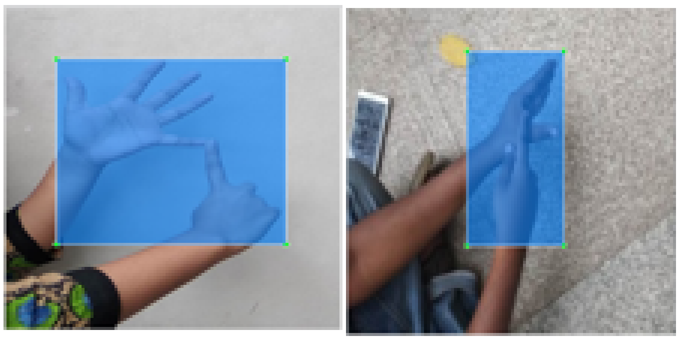


Figure 3.5: Images with bounding box.

For some specific letters, we have collected above 500 images. In total, we collected a total of about 2000 images.

Table 3.1: **BdSLImset** Description.

Total Images	**Total Class**	**Images/ Class**	**Image Size**	**Image Resolution**	**Image Background**	**No. Of Person Participated**	**Training Set : Test Set**
2000	10	200 [1]	<=200kb	<= 700*1280	Dynamic	20	8:2

3.1.3 Other BdSL Datasets & BdSLImSet

Most of the existing BdSL detection models dataset do not have variation in background and lighting conditions. Most datasets are neither open source for future research domain. A comparison between datasets used in previous works and our dataset is shown in Table 3.3.

3.2 BdSLVidset

Recognition of sign words falls under the domain of action recognition, which means recognizing frames in sequence for each word. For word recognition, we needed a video dataset.

[1]'ই' and 'এ' have about 500 images in each class.

Table 3.2: BdSL sign letters in *BdSLImset* and corresponding ‘ID’.

Id	**Gesture**
1	oo (অ)
2	a (এ)
3	i (ই)
4	e (ঈ)
5	u (উ)
6	k (ক)
7	kh (খ)
8	ga (গ)
9	dh (ঢ)
10	o (ও)

Table 3.3: Comparisons between datasets used in previous work and in our method.

Related works	**Background**	**Image Per Class × Total Classes**	**No. of Signers**	**Available**
Rahman et al.2014 [20]	Static	36 × 10	10	×
Rahman et al. 2015 [21]	Static	10 × 10	10	×
Ahmed et al. [18]	Static	37 × 14	3	×
Our (BdSLImset)	Randomized	10 × 10	10	√

As no previous research is done on BdSL word recognition using deep learning, there are no available datasets. We have generated our own *BdSLVidset* for our system. An example of BdSL word 'You' as a sequence of images is given in Figure 3.6.

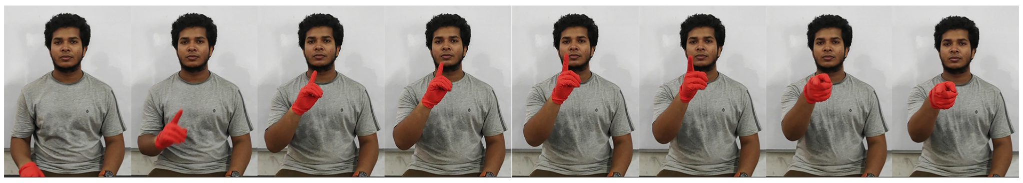


Figure 3.6: The sequence of images of a video gesture belonging to class 'Tumi (You)'.

In Figure 3.7, it can be noticed that only with the variation of palm movement the words are recognizable. So for the purpose of recognizing the position of the hand, we can simply remove all the background and only track the palm movement.

3.2.1 Bangladeshi Sign Language Video Dataset - *BdSLVidset*

BdSLVidset is inspired by the dataset of the Argentine Sign Language Dataset [14]. Currently, our *BdSLVidset* dataset has 4 different labeled sign letters. We collected about 50 videos for each gesture and 5 subjects have participated. The dataset is divided into a training set and a testing set with a ratio of 6:4. An overall review of BdSLVidset has been given in Table 3.4 and the words with corresponding 'ID' is shown in Table 3.5.

Table 3.4: **BdSLVidset** Description.

Total Videos	**Total Class**	**Videos/ Class**	**Video Size**	**Video Background**	**No. Of Person Participated**	**Training Set : Test Set**
200	4	50	<=800kb	White with lighting variation	5	6:4

3.2.2 Recordings

The database was recorded in two sets. The first recording was done in an outdoor environment with natural lighting. While the second recording was done in an indoor environment with artificial lighting, to provide differences in illumination between signs. In both sets

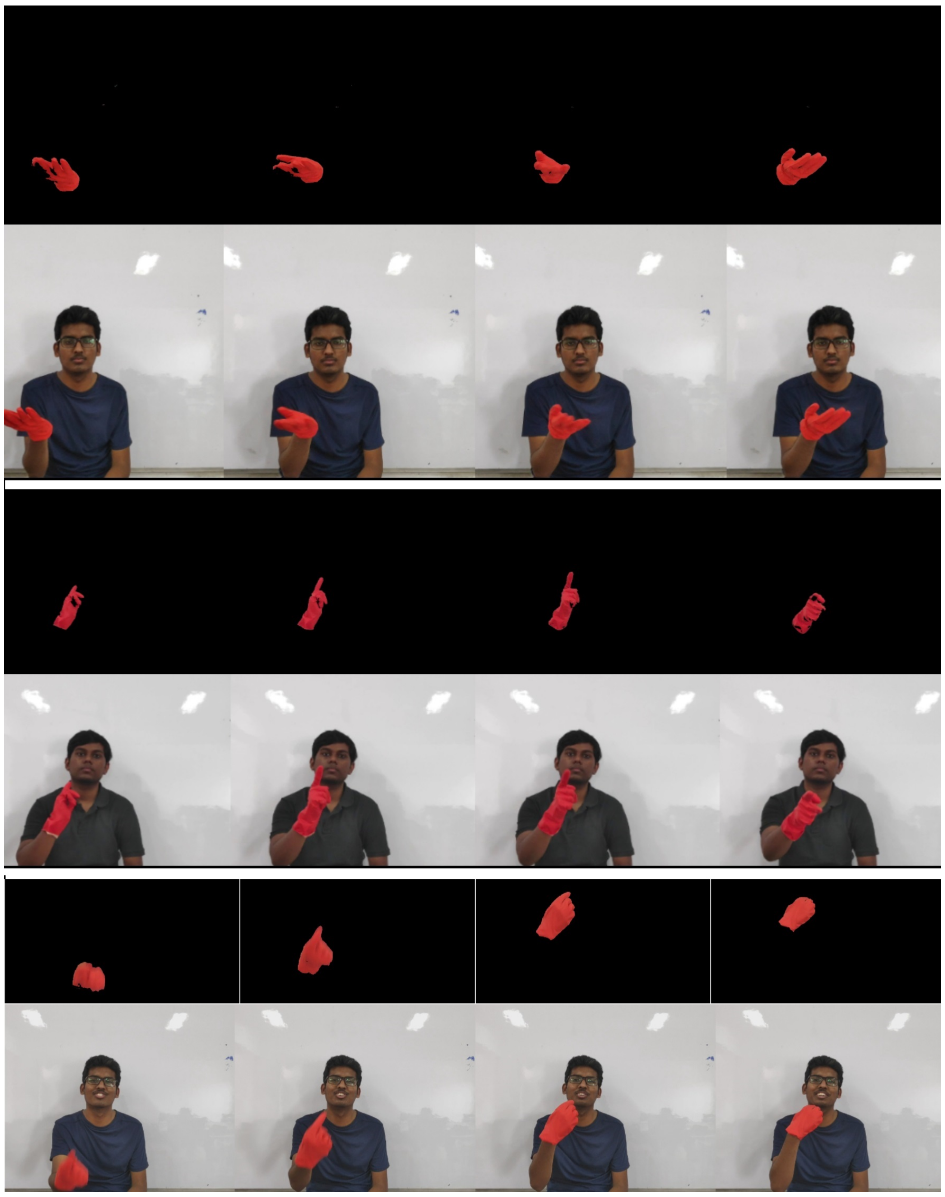


Figure 3.7: Samples before and after background removal of class 'Kemon', 'Tumi' and 'Ami' from top to bottom.

of recordings, subjects wore clothes having a different color than the hand gloves and performed the signs standing or sitting, with a white wall as a background. To simplify the problem of hand segmentation within an image, subjects wore red-colored gloves. These substantially simplify the problem of recognizing the position of the hand and performing its segmentation and remove all issues associated with skin color variations, while fully retaining the difficulty of recognizing the hand-shape. Each sign was executed imposing few constraints on the subjects to increase diversity and realism in the database. All subjects were right-handed non-signers and were taught how to perform the signs during the recording session by showing them a video of the signs as well as performed by one of the authors. All subjects practiced each sign a few times before recording.

Table 3.5: BdSL sign words in *BdSLVidset* and corresponding id.

ID	Gesture
1	Tumi
2	Ami
3	Kemon Acho
4	Valo Achi

The camera employed was the same in both sets of recording (Xiaomi POCOPHONE). The camera was placed 2m away from the wall. The resolution of the videos is 1080×2246, at 30 frames per second.

3.2.3 Pre-Processings

In the video dataset, each video was temporally segmented so that the frames in the beginning or end of the video with no movement in the hands were removed. After that, each video was divided into 40 frames and converted to gray-scale images (see Figure 3.8 and Figure 3.9).

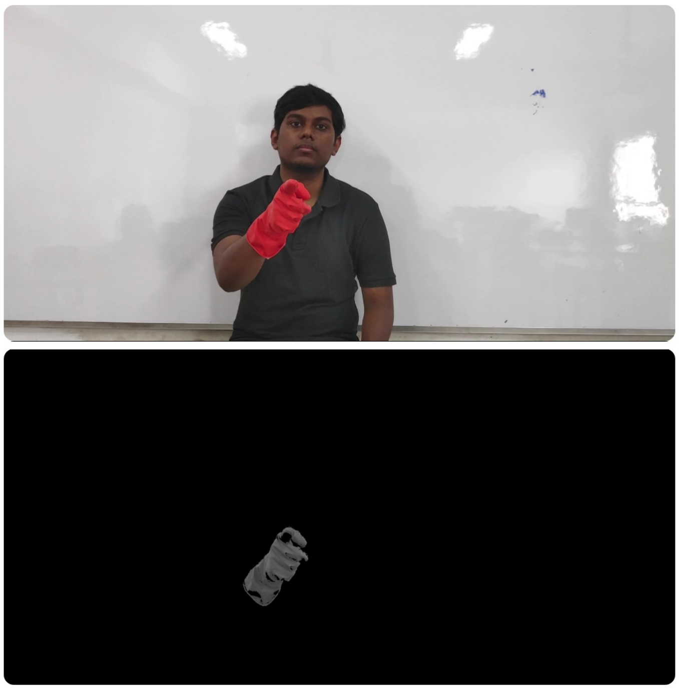


Figure 3.8: Final sample after processing and background removal of a frame.

3.2.4 Limitations

- A simple colored glove is needed to segment the hand portion from everything else.

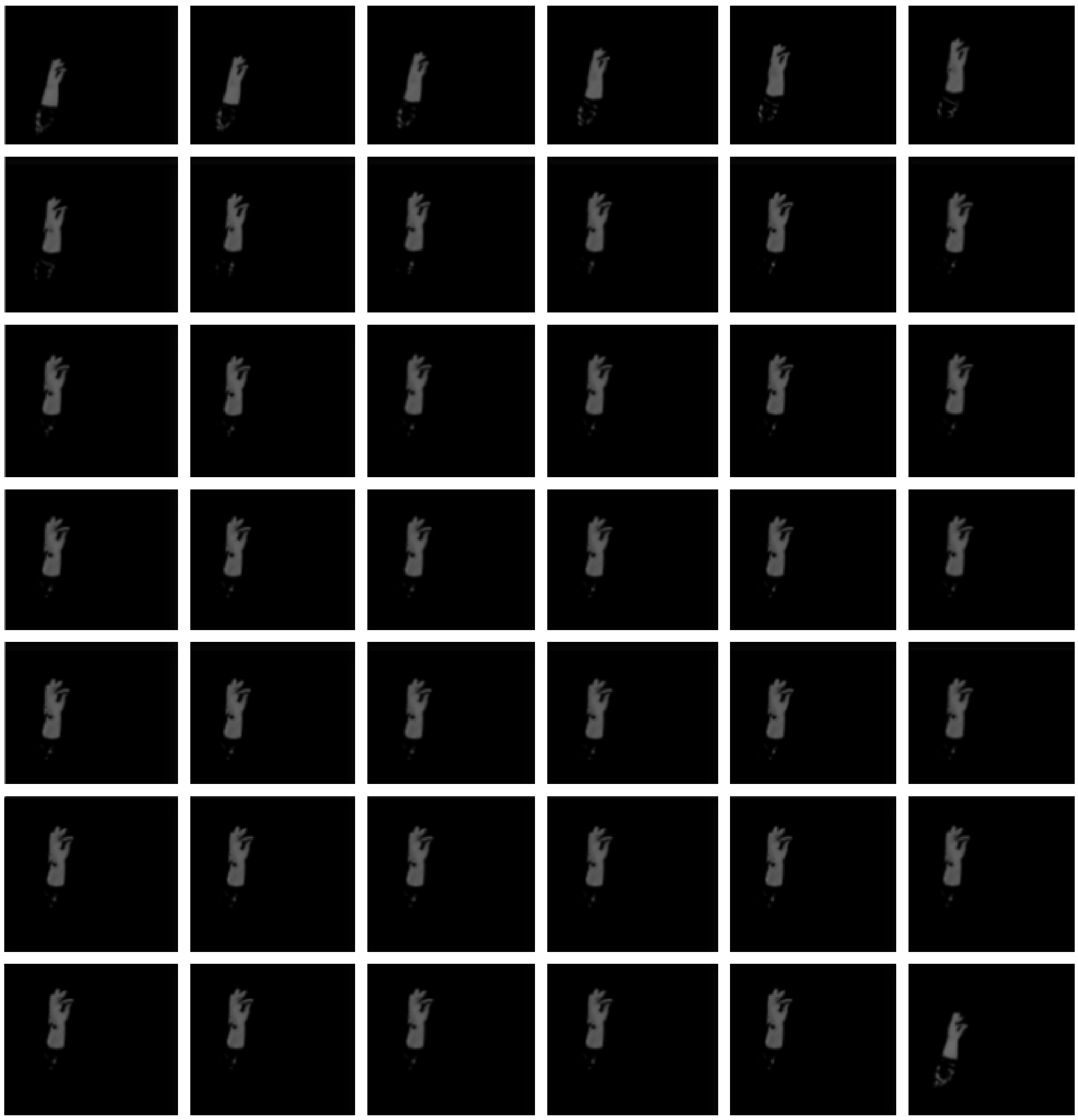


Figure 3.9: 40 frames belonging to a video sample after processing and background removal.

- The background must not contain any similar color to the hand glove.
- Currently, dataset is prepared for only four words.
- Less variation in the background for the system simplicity purpose.
- Our character recognition and word recognition are two different systems and the work done are completely separated.

3.3 Summary

In this chapter, we have demonstrated the specifications of our datasets – *BdSLImset* & BdSLVidset – which have been built by ourselves for our systems. *BdSLImset* consists of a total of 2000 images and we have used 10 participants. For *BdSLVidset* we have collected 200 videos and used 5 participants for data collection.

Chapter 4

Proposed Methodology

In one of our works, we emphasize identifying and recognizing sign letters in images that can be categorized as an object detection problem. On the other hand, identifying and recognizing sign words from video frames falls under the domain of action recognition problem. Our study on real-time object detection leads us to pivot on Faster R-CNN and for action recognition, RNN based LSTM model is best suited. Therefore, we implemented both of the two methodologies individually to get the best result from to help the hearing-impaired community on our own datasets. In this chapter, we only review the essential steps of using these models in our systems. For more details on the architectural design of the models, section 2.1.4 and 2.1.5 and for detailed experimental phase and results on our methodologies, chapter 5 is referred to the reader.

4.1 Sign Letter Recognition

For Sign Letter recognition, we have used the Faster R-CNN Inception V2 model (section 2.1.4) to train our system. In this section, we provide an overview of our sign letter recognition system starting with input, followed by how the training and testing phase were designed and also an overview of the real-time user testing phase through our system *Bd-SLImset*.

Dataset Preparation:

While investigating, we found lacking a proper dataset to train a neural network. As we want to develop a real-time system our classifier must be stronger; thus the dataset must be adequate. Therefore, we focus on developing a dataset and determine the following criteria for our training images.

1. There must be variations in gestures for the signs of the same type. Images contain

desired gesture of letter which is partially obscured, overlapped with something else, or only halfway in the picture

2. Random backgrounds and lighting conditions must be considered.

The above criteria were missing in the previous datasets by different authors and we kept these in our considerations while collecting training images. A detailed analysis of different datasets is also provided in our **Dataset** chapter 3.

Training Phase Input:

Training phase inputs are images from *BdSLImset* dataset. Each image contains additional information of it's labeled class, bounding box values. A batch of 100 images was fed into the network in each iteration to train. Data augmentation has been done on the input images before training by the system to make the dataset more robust.

Training Phase:

As mentioned earlier in the overview of Faster R-CNN model architecture in Section 2.1.4, our system breaks an image into different regions of different anchor size. We have used three different sizes of anchors of 3 different ratios (see Figure 5.5). The system tries to identify if there is an object that exists in each region. If the system finds the object it keeps the anchor for further training to make the convolutional feature map else discard the region (see Figure 5.6). Then the feature map goes to the RPN and proposes regions with the probability of that region being a sign gesture of a letter. Figure 4.1 shows the entire working process of the network. Here, RoI pooling resizes the feature map into fixed sizes for each proposal in order to classify them into a fixed number of classes.

The system learns from the features with a forward pass and calculates the loss in each iteration. And through back propagation, the system updates the weight in each layer accordingly. This process repeats until the loss is under 0.03. For classification sparse cross-entropy is used and for bounding boxes, Smooth L1 Loss is used. When the loss is minimal, the system gets ready to identify signs in images.

Test Phase:

Our dataset is divided into train and test sets. The train set has been used in training and the test set identifies how much our model has actually learned. In this phase, the only image is passed to the network unlike the train phase with bounding box information. Dividing images into several regions, the system identifies the signs and gives a prediction. Firstly, the system tries to identify which region has a sign with the help of the RPN layer and if any region contains any sign, the system generates bounding box information and predicts its class. Test set accuracy is calculated by matching the true label and the predicted label of each image. The test set accuracy of our system is 0.941. Chapter 5 has elaborate description on our test set results.

Real-Time Testing:

We have implemented a real-time system that is user-friendly and predicts from real-time video frames. 30 frames per second are captured from a real-time video and the system predicts the frames and shows continuous output in the user display interface(see Figure 4.3). It takes no time to recognize and also can identify the change in signs by user instantly.

Flow diagram of Figure 4.4 visualizes our overall BdSL letter recognition system.

4.2 Sign Word Recognition

Sign words contain a sequence of action. Only predicting on one image is not enough, it needs prediction on multiple frames and to combine the predictions together to identify the class. The general idea behind sign word recognition is, we first apply CNN-based model to get feature values, just like sign letter recognition, from each frame. Then we combine these values sequentially through an RNN based LSTM model. LSTM models take each frame as input and also keeps the outputs of previous inputs and actually combines them to make a final prediction.

We have used two methods to train the model on the temporal and the spatial features. Both methods differ by the inputs given to RNN to train it on the temporal features.

4.2.1 Dataset Used

As mentioned earlier, action recognition needs prediction on multiple frames at one time. It gets more complicated than just recognizing just a sign letter. Unlike sign letter recognition system which is trained on a robust dataset, sign word recognition system datasets is more processed for simplification purpose. *BdSLVidset* dataset has been used to train our model. A detailed explanation of our dataset is given in Chapter 3.

4.2.2 Our RNN Model

We created an RNN model based on LSTMs. Its input size varies for our different methodologies, explained in later sections. It has 256 LSTM units in the middle followed by a fully connected softmax layer which size is the same as the number of classes. The last layer is a regression layer or the output layer (see Figure 4.5).

4.2.3 Method I

In this approach, we extracted spatial features for individual frames using the inception model (CNN) and temporal features using RNN. Each video (a sequence of frames) was then represented by a sequence of predictions made by CNN for each of the individual frames. This sequence of predictions was given as input to the LSTM (see Section 2).

- First, we will extract the frames from the multiple video sequences of each gesture (see examples in Section 5).
- After the first step, noise from the frames i.e background, body parts other than hand are removed to extract more relevant features from the frame.
- Frames of the train data are given to the CNN model for training on the spatial features. We have used the inception model for this purpose which is a deep neural net (see Figure 4.6).
- Store the train and test frame predictions. We will use the model obtained above step for the prediction of frames.
- The predictions of the train data are now given to the RNN model for training on the temporal features. We have used the LSTM model for this purpose.

4.2.4 Method II

In this approach, we have used CNN to train the model on the spatial features and have given the output of the pool layer, before it's made into a prediction, to the RNN. The pool layer gives us a 2048 dimensional vector that represents the convoluted features of the image, but not a class prediction. The rest of the steps of this approach are the same as that of the first approach. Both approaches only differ by inputs given to RNN.

Real-Time Testing:

We have implemented a real-time system that is user-friendly and predicts from real-time video frames. 40 frames per second are captured from a real-time video. The user has to input the time duration of the action in the recorded video and after a while, system outputs the result. The user has to wear a normal red hand-glove and the background should not contain any similar color background to the hand-glove. For more details on result section, reader is reffered to section 5.2.2.2.

Flow diagram of Figure 4.7 visualizes our overall BdSL word recognition system.

4.3 Summary

In this chapter, we have described our systems architectures elaborately. This section is subdivided into two other sections - one for letter based recognition system and another is for word based recognition system. Both of them contains the processes of our systems evaluation step by step with necessary description, figures and flowcharts.

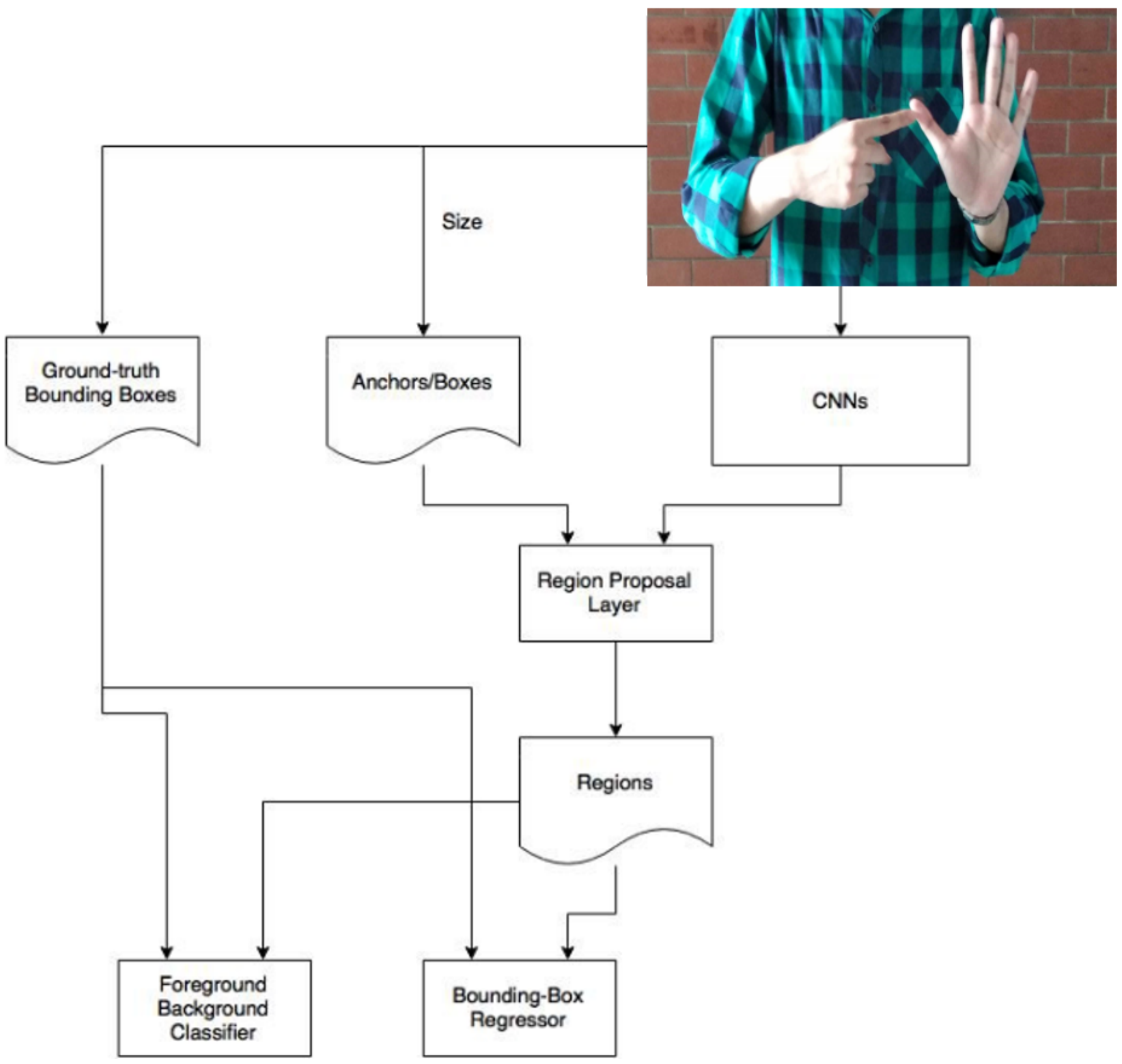
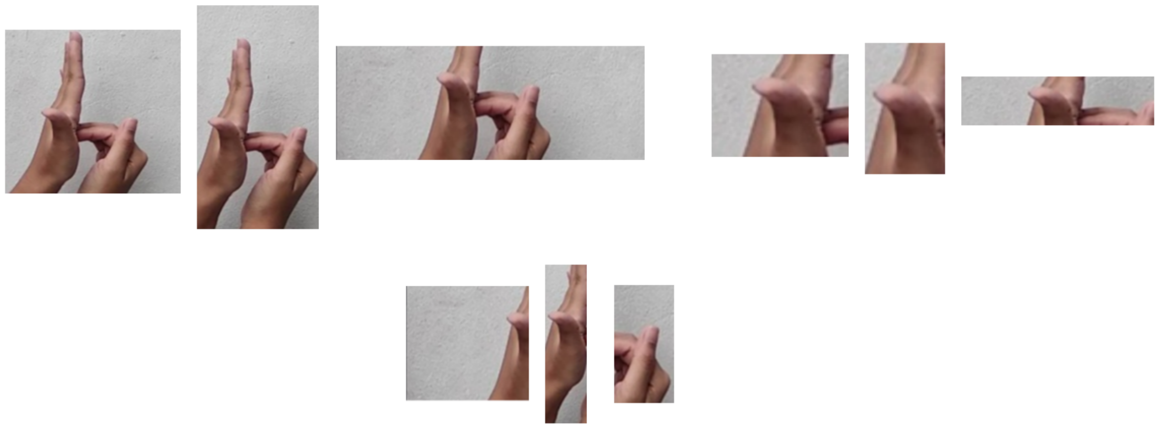
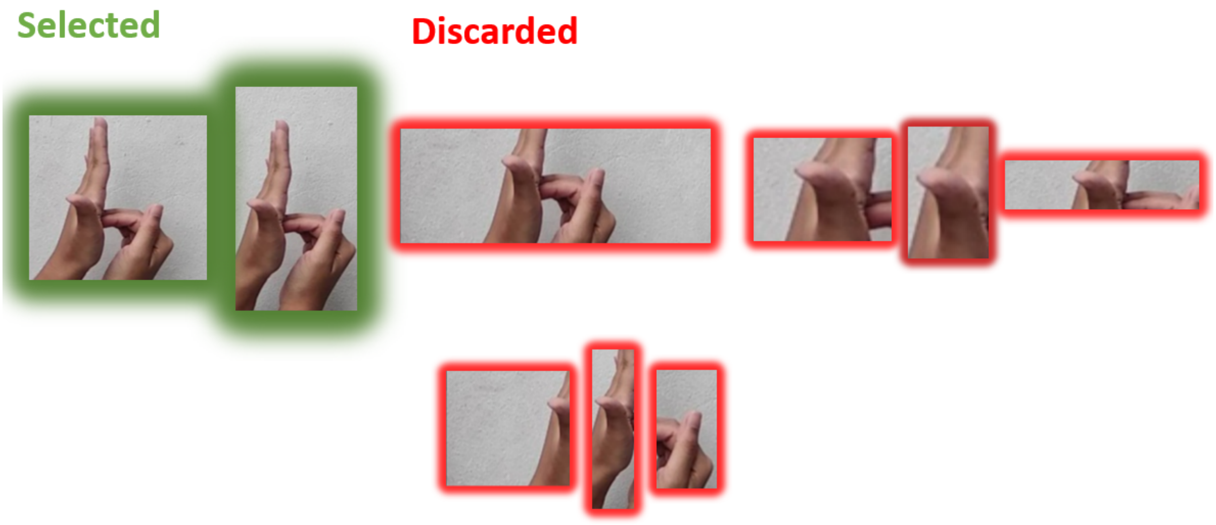


Figure 4.1: Detailed architecture of the network. Firstly the input image goes into the CNN framework and creates a feature map. RPN proposes anchors with higher probability of being an anchor and the RoI pooling classification is performed at last to finalize the classification.

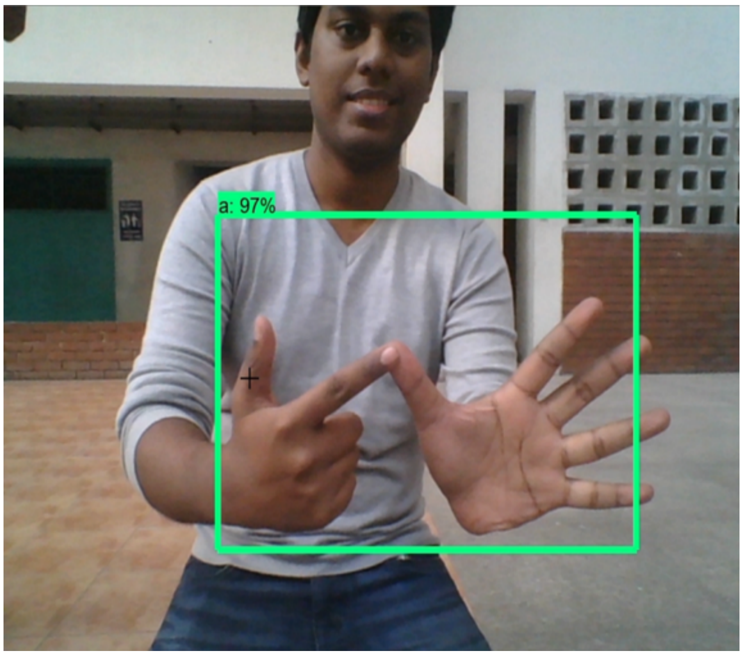


(a) Sub-diving an image with different sizes and ratios of anchors.



(b) Discarding the regions that does not contain any sign letter.

Figure 4.2: Example of anchor boxes of 3 different sizes and shapes. Anchors with three scales or sizes, 128x128, 256x256, 512x512 and each of the the three boxes have height width ratios 1:1, 1:2 and 2:1 respectively.



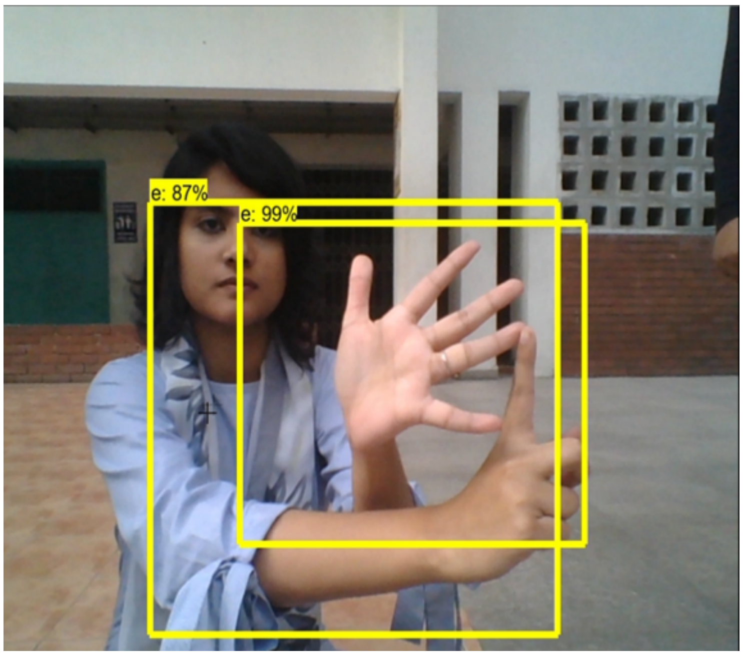


Figure 4.3: Sample of our real-time systems user interface output. It takes no time to detect the signs and even the system can also differentiate the simple variations in signs easily without any background barrier.

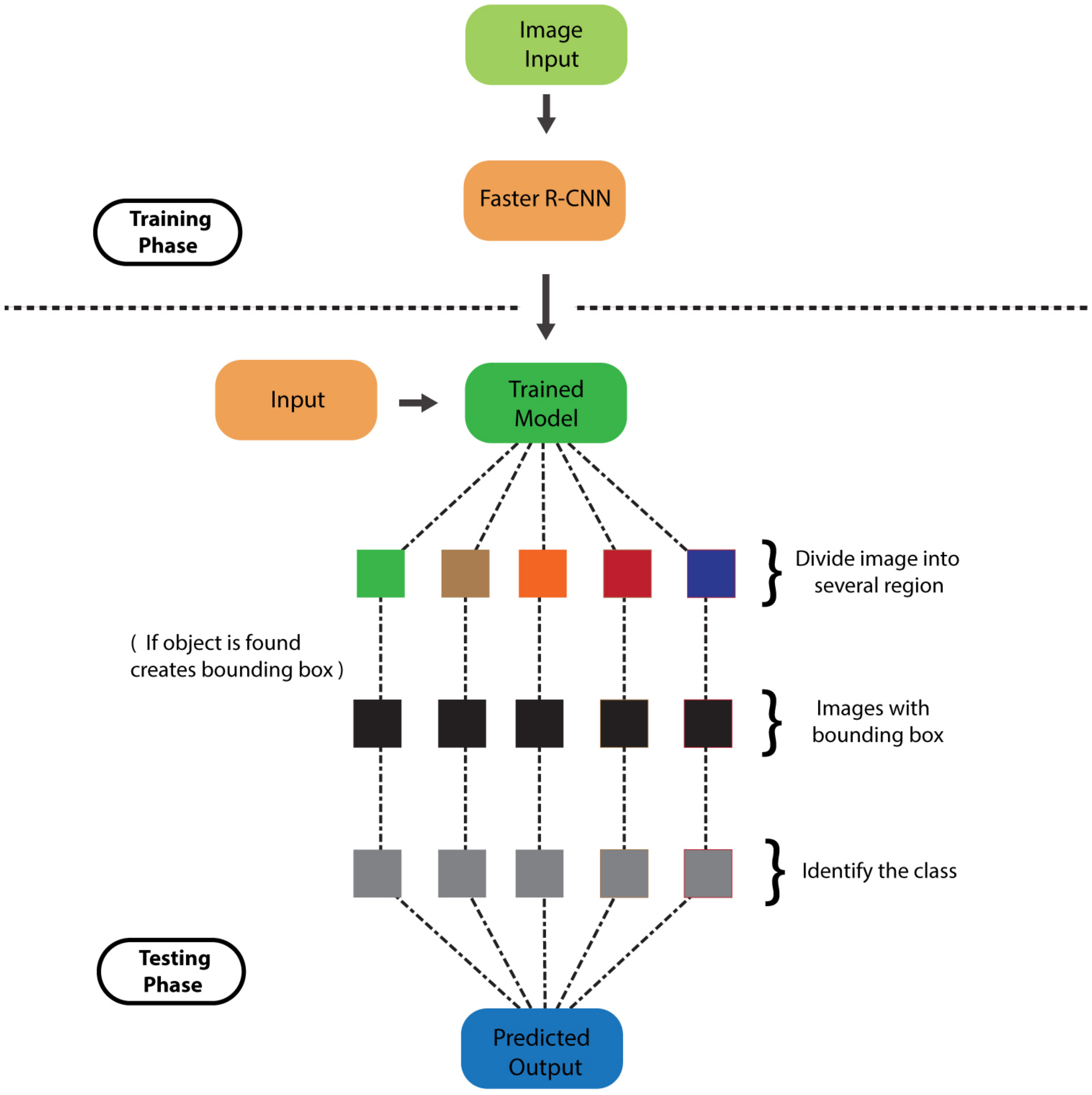


Figure 4.4: Flow diagram of our BdSL sign letter recognition model. Inputs(collected from *BdSLImset* are fed into the Faster R-CNN model. Each iteration a batch size of 100 images is passed to the training system. After training, images from the test dataset are randomly passed to the trained model. The trained model, search for signs in the image by subdividing the image into various regions. Finally, passes an output with a probable class label and bounding box area information.

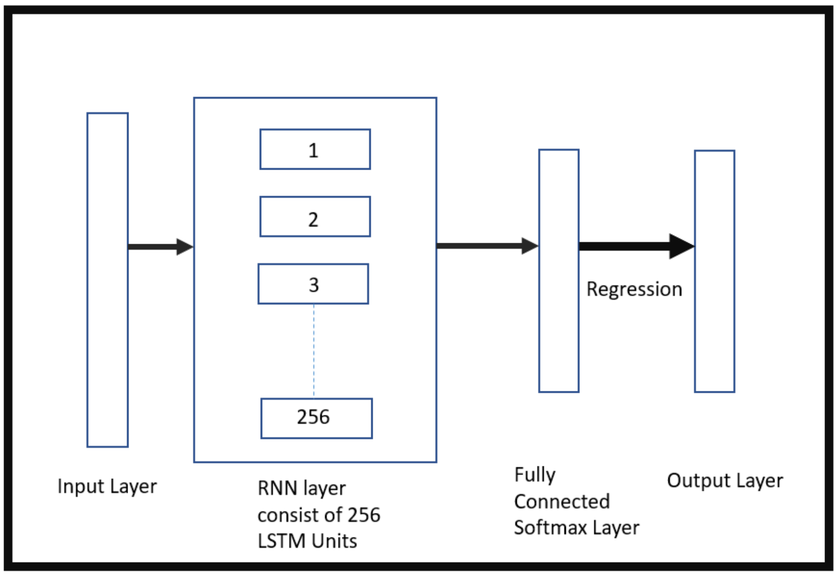

1
2
3
256
Regression
Input Layer
RNN layer consist of 256 LSTM Units
Fully Connected Softmax Layer
Output Layer

Figure 4.5: RNN model with a input layer, followed by a LSTM layer with 256 units, a fully connected softmax layer and a output layer.

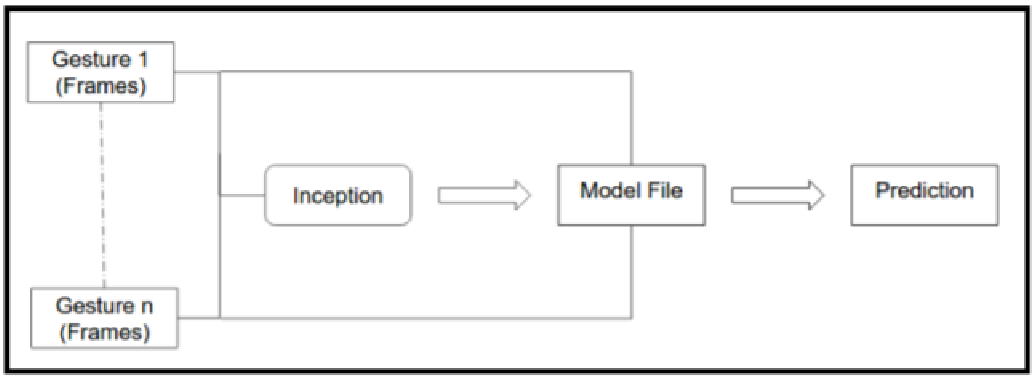


Figure 4.6: Training on CNN of spatial features and prediction. Frames of each gesture are passed to a CNN-based inception model. This model generates predictions on each frame for every video of gestures. [8]

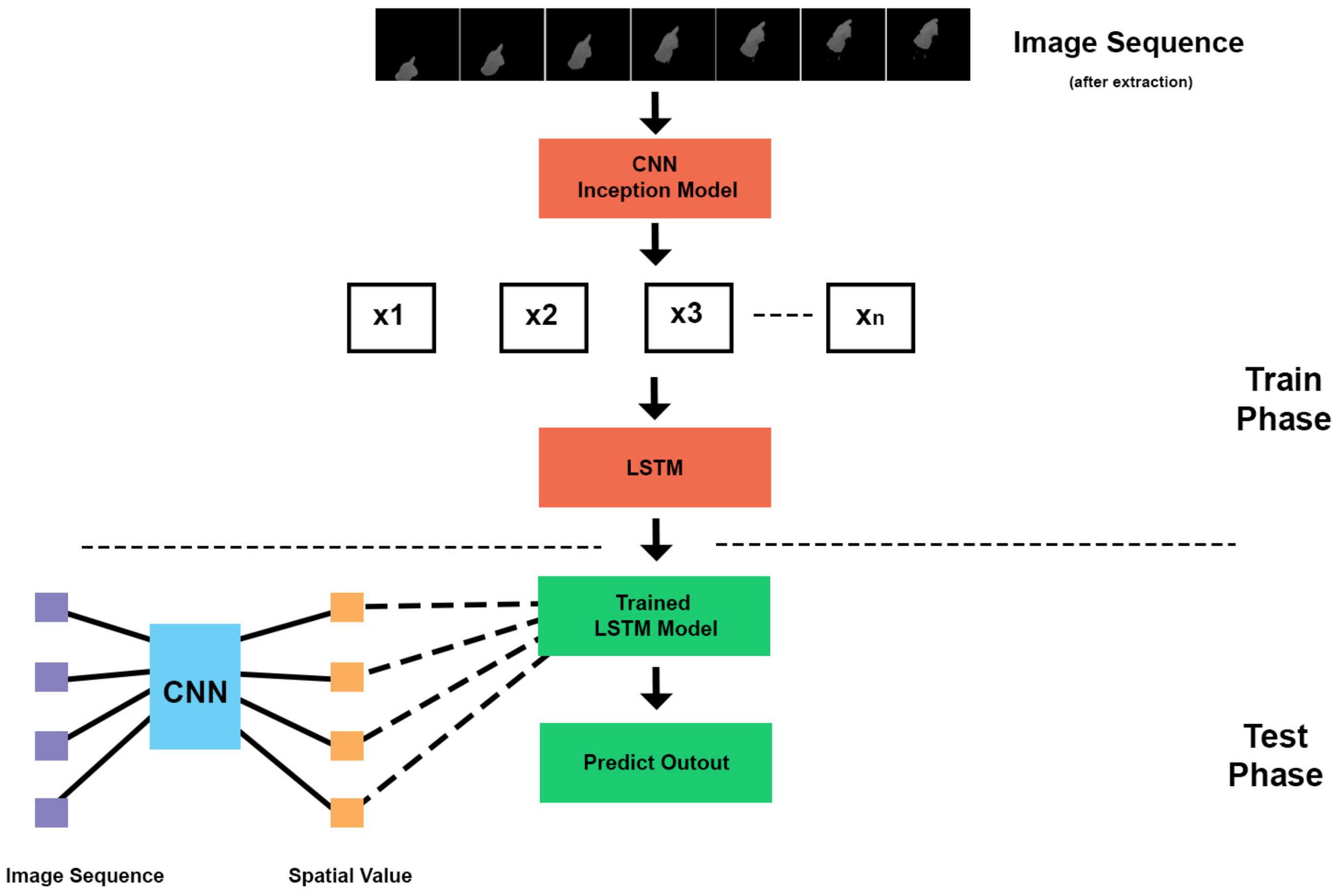


Figure 4.7: Flow diagram of our BdSL sign word recognition model. Frames of each gesture are passed to a CNN-Inception model. The generated values from this step is passed to the LSTM model as input. The final training is done on the LSTM model. Test phase goes through the same steps but in this phase, the system predicts an output for a video-based learning from the training phase.

Chapter 5

Experimental Design and Results

This section represents the experimental results of our systems implemented on our datasets. It is divided into two subsections: **Sign Character Recognition** and **Sign Word Recognition**. We start with the experimental setup followed by the experimental phases and the final outcomes of the systems.

5.1 BdSL - Sign Character Recognition

This section represents the experimental result of our proposed BdSL letter recognition with Faster R-CNN model on the prepared *BdSLImset* dataset. We start with experiment phases followed by the final outcome of the system.

5.1.1 Experimental Setup

Our techniques are implemented in `Tensoflow-GPU V1.5` and `cuda V9.0`. The training was performed by adopting the `Faster RCNN Inception V2` model. The experiments have been conducted on a machine having CPU from Intel ®. Core^{TM} $i7-$ 7500U of 2.7 GHz, GPU Nvidia 940mx with 4.00 GB and with 8.00GB memory on a Windows 10 operating system.

5.1.2 Experimental Phase

We have used different versions of ***BdSLImset*** dataset and also two models on the final version of the dataset we generated. We will describe the results for the training and testing set images from our datasets and also the application level by the users in real time.

5.1.2.1 Method I

Dataset Used: *BdSLImset* Dataset (Version—1).
It had a much lesser number of images per class and fewer variation in the dataset (see Table 5.1).
Model Applied: Faster R-CNN

Output: The output of this approach has been described in Table 5.2

Table 5.1: ***BdSLImset-*** (Version—1) dataset description.

Total Class	10
Number Of Images per Class	50

Table 5.2: Results of first approach

Training Time	8 hours
Average Accuracy (On Test Set)	0.8
Detection Accuracy (In Real-time)	Faulty

Comments:

1. The results are not satisfactory, both on the test set and application level.
2. Need more dataset to train the model as the sign letters have many similarities among themselves. So it is hard for the computer vision system to recognize.

5.1.2.2 Method II

Dataset Used: *BdSLImset* Dataset (Final Version).
After our initial experiment, we generated a much larger dataset for our classifier system and tried a different model to build a robust classifier. The elaborate dataset description is given in Chapter 3.

Model Applied: Faster R-CNN , YOLO- Inception V2 Model

Output: The comparison between the outcome of the two applied models have been described in Table 5.3.

Comments:

1. Faster R-CNN model outperformed YOLO Inception V2 on our dataset.

2. An increased dataset helped the classifier system to learn about the features more accurately.

3. But for YOLO Inception V2 Model, this version of our dataset is still not enough. YOLO is claimed to be less calculational expensive and more robust as a model. But it has more hidden layers, need more powerful GPU and processing power and a larger dataset to get trained on. YOLO models are also compatible with mobile applications also.

Table 5.3: Comparisons between two applied models on *BdSLImset*.

Models	**Faster R-CNN**	**YOLO Inception V2 Model**
Training Time	12 hours	8 hours
Average Accuracy (On Test Set)	0.9415	0.6
Detection Accuracy (On Application Level)	Accurate with average of 90% confidence rate	Faulty

As from all the above results, we can finally come to the conclusion that Faster R-CNN The inception model on BdSLImset has really performed well. It has served our purpose of building a robust classifier for the sign letter recognition system. The elaborate experimental results on the Faster R-CNN model have been described in the further sections.

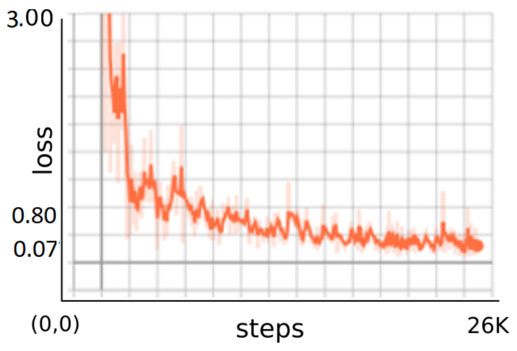
5.1.3 Experimental Result

We split this section into two other subsections. At first, we will describe the training phase. In the latter part, we will discuss our results.

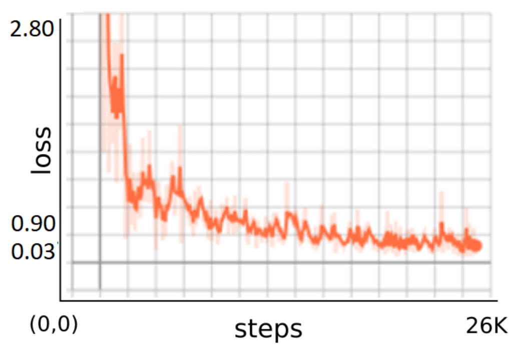
5.1.3.1 Training

The initial training was for 10 classes with 50 images for each letter in the dataset model. It took about 8 hours and 28000 iterations to train the model. Training started with a loss of 3.00 and quickly dropped to 0.8. We stopped our training when the loss became constant at 0.07 (see Figure 5.1a).

We increased our dataset with more images for all the classes. This dataset contains 200 images for each class. And then we trained with our model again. It took about 12 hours to train and training started with a loss of 2.8 and quickly dropped to 0.9. We stopped our training when the loss became constant at 0.03 (see Figure 5.1b).



(a) First Approach



(b) Second Approach

Figure 5.1: Tensor Board loss graph.

5.1.3.2 Results on Test Dataset

For initial training, the result was not satisfactory. As the sign letters have many similarities among them, the detection were not accurate for all letters especially for the letters with similar hand gestures. But later it gave an accurate result with an average of 90% confidence rate.

In our later training with the increased dataset, we got an extraordinary result. The accuracy was higher and it was detecting all the letters almost perfectly. The results of Table 5.4 are based on the Faster R-CNN model classifier on our final version of BdSLImset dataset. The table shows the accuracy of each class on the test dataset. On testing time, the images from testing were given randomly.

Table 5.4: Results on the letters with accuracy.

Id	Gesture	No. Of Image In Test Set	No. Of Correct Classification	Accuracy(%)
0	oo (অ)	40	38	95
2	a (এ)	40	40	100
3	i (ই)	40	35	87
4	e (হ)	40	35	87
5	u (উ)	40	30	75
6	k (ক)	40	40	100
7	kh (খ)	40	40	100
8	ga (গ)	40	40	100
9	dh (চ)	40	40	100
10	o (ও)	40	39	97.5

5.1.3.3 Results On Real Time System by Users

We have implemented a real-time system to test our classifier model by the users. We tested our trained model in various environments and with different persons and we got a satisfactory result. We tested our system on various backgrounds by different users (see Figure 5.2). The confidence rate was 98% on average.

5.2 BdSL - Sign Word Recognition

We have implemented two different models using our *BdSLVidset* dataset. We describe this part both on the training and testing videos that are available in our datasets and also on application level by the user in real time.

5.2.1 Experimental Setup

Our techniques are implemented in `Tensoflow-GPU V1.5` and `cuda V9.0`. The training was performed by adopting the `Faster RCNN Inception V2` model. The experiments have been conducted on a machine having CPU from Intel ®. Core^{TM} $i7-$ 7500U of 2.7 GHz, GPU Nvidia 940mx with 4.00 GB and with 8.00GB memory on a Windows 10 operating system.

5.2.2 Experimental Phase

We have implemented two different models using our *BdSLVidset* dataset. We describe this part both on the training and testing videos that are available in our datasets and also on application level by the user in real time.

5.2.2.1 Method I

In this approach, we divided each video of gestures into 40 frames. For each frame spatial feature is extracted using Deep Neural Network Inception Model and temporal features using RNN. Each video was then represented by a sequence of predictions made by CNN model for each individual frames and this sequence of predictions was given as input to the RNN.

Step 1:
Extracting frames from the video sequences (see Figure 5.3). And the image frames go through necessary processing (described in Chapter 3).

Step 2:
The frames of train data are then fed into the Deep Neural Net Inception model for extracting the spatial feature of each image. This returns a list of values of the probability of the frame belonging to each class (see Table. 5.5). As we have 4 class in our system, so we get 4 class probabilities. These values are stored in a .pkl file.

Frames	**Spatial Feature Value (Class Predictions) Extracted By CNN:**
Frame 1:	[[0.9968306422233582, 0.002878435654565692, 0.0002908589376602322, 0.0068306422233582], 'Ami']
Frame 2:	[[0.9968306422233582, 0.002878435654565692, 0.000290858937232266, 0.00306422233576], 'Ami']

Table 5.5: The given illustration is of how each frame is assigned a spatial value by CNN.

Step 3:
This stored values are passed into the RNN (LSTM model) for training on the temporal features (see Figure 5.4).

5.2.2.2 Method II

The first and third step is the same in the second approach but the RNN is not trained with class probabilities of each frame rather the convoluted features of each frame is feed into the RNN (see Table. 5.6). These values are stored in a .pkl file. The CNN model returns the output of the pool layer, not the output layer and which is a 2048 dimensional vector.

Comments:

1. The result is satisfactory for both models but as we have fewer classes. The actions do not have many similarities among them and we have a pretty large dataset. It may vary when the number of classes increases.
2. Approach 1 has some drawbacks, although it is faster in the training phase and less computational expensive but when the number of classes increases the time need for each prediction increases. The length of a probabilistic prediction by CNN in the sequences of predictions of frames is equal to the number of classes to be classified.
3. But for Approach 2, faster in this case as the RNN's input layers do not depend on the number of classes and must give a higher accuracy as well even when the number of classes is increased.

Frames	**Spatial Feature Value (Pool Layer Output) Extracted By CNN:**
Frame 1:	[[0.2489597201347351, 0.07230774313211441, 0.27201735973358154, 0.27852416038513184, 0.028210513293743134, 0.5794256329536438, ...],'Ami']
Frame 2:	[[0.2489597201347351, 0.07230774313211441, 0.27201735973358154, 0.27852416038513184, 0.028210513293743134, 0.5794256329536438, ...],'Ami']

Table 5.6: The given illustration is of how each frame is assigned a spatial value by CNN.

Table 5.7: Sample Table

ID	Gesture	Video	Classification		Accuracy (%)	
			Approach1	Approach2	Approach1	Approach2
1	Ami	20	20	20	100	100
2	Tumi	20	20	20	100	100
3	Kemon_Acho	20	20	20	100	100
4	Valo_Achi	20	19	20	100	95

5.2.2.3 On Real Time System by Users

Our system records the video from users and the user must specify the duration of gestures by giving input of start and end time of which portion of the video contains sign, for system to predict from that real-time video. Our system can predict multiple words at a given time duration. Our system is tested for both of our two methodologies. Comparisons between the time of prediction for both approaches is shown in Table 5.8.
We tested our system on various backgrounds by different users. The Figures in 5.5, 5.6 and 5.7 demonstrate the results of the real-time recognition.

Comments:

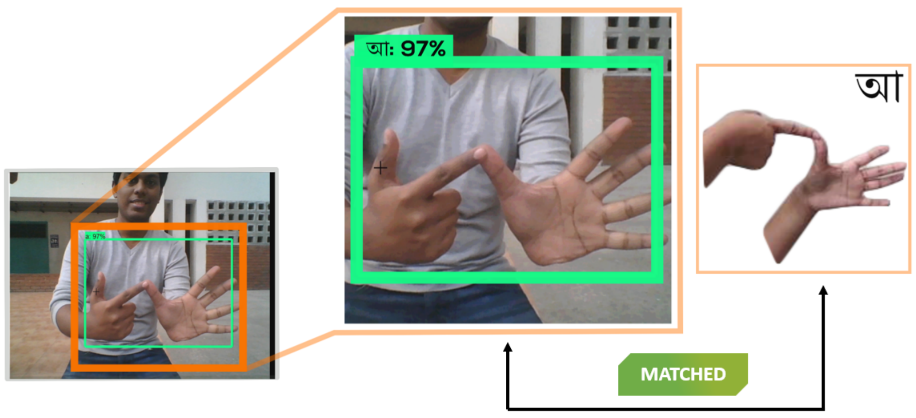
1. Approach 2 based system outperforms the Approach 1 based system in every way. Convoluted features rather than class prediction probability seem to be much more efficient in training an RNN model for gesture recognition.

Table 5.8: Prediction comparison between different approaches.

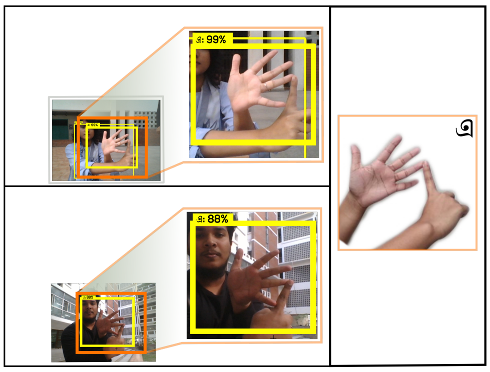
Approaches	Time Needed To Detect Each Gesture	Multiple Gesture at a time
Approach 1	About 1.30 minutes	No
Approach 2	About 30 seconds	Yes

5.3 Summary

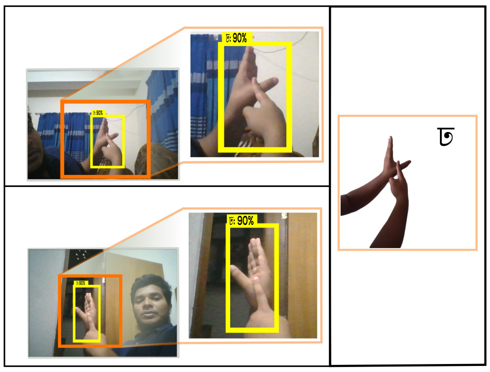
In this section, we have described the results of our systems step by step with necessary figures and tables. We have used different methods and architectures for our system. Each of the results are described separately in this section.



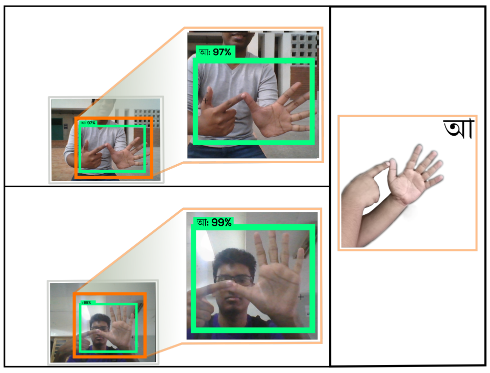
(a)



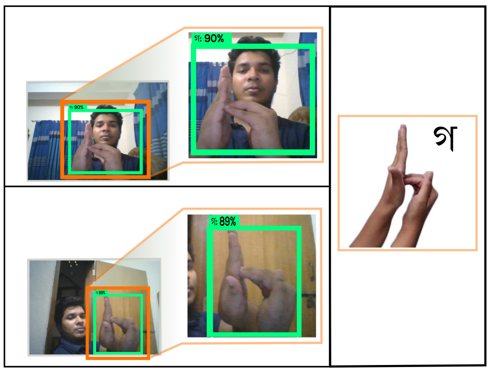
(b)



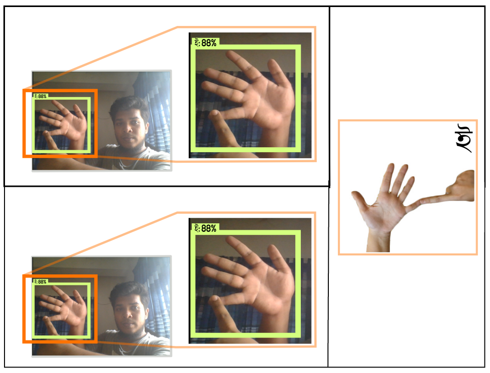
(c)



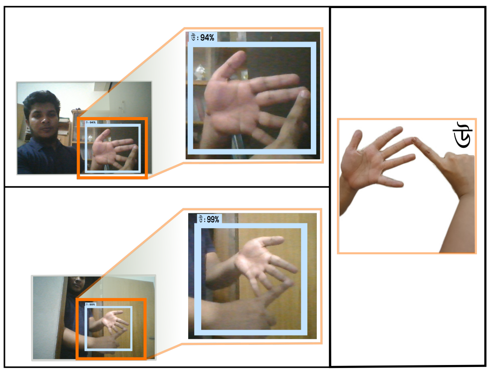
(d)



(e)



(f)



(g)

Figure 5.2: Some results of real time detection. This experiment had been done on various situation, i.e. different background, illumination and angles. The 1st column shows the detection and the 2nd column has a zoom version of detected portion for better investigation. A reference sign is included at the rightmost column as ground truth collected from *BdSLImSet*.

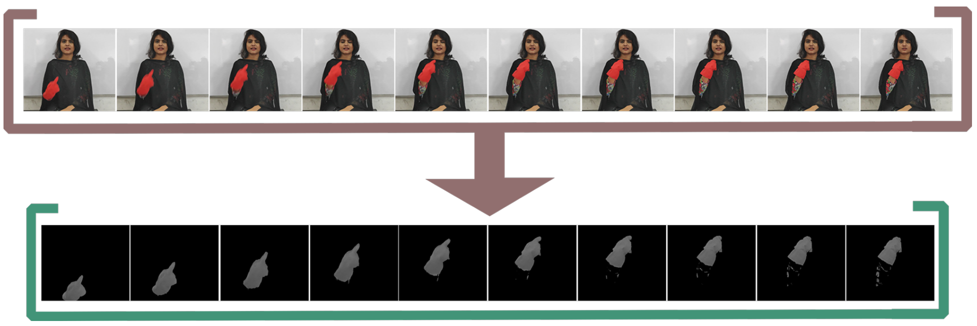


Figure 5.3: Pre - processing on video dataset. Backgrounds are removed and each frame is converted to gray-scale image.

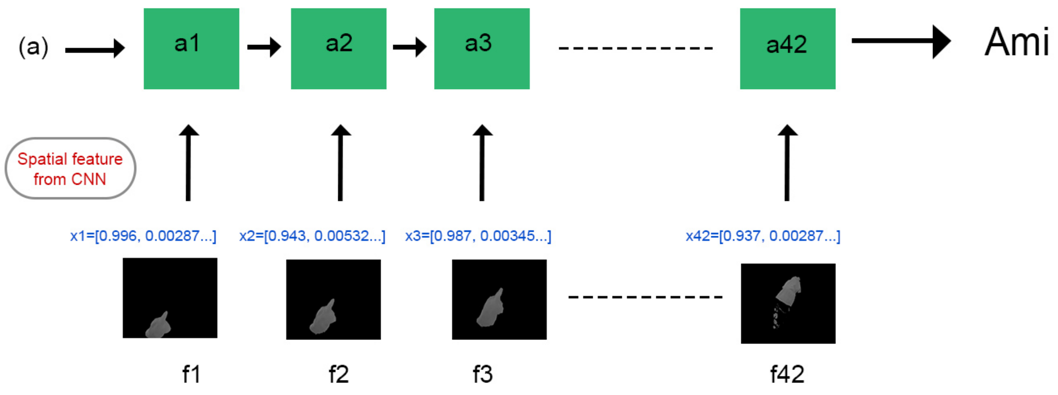
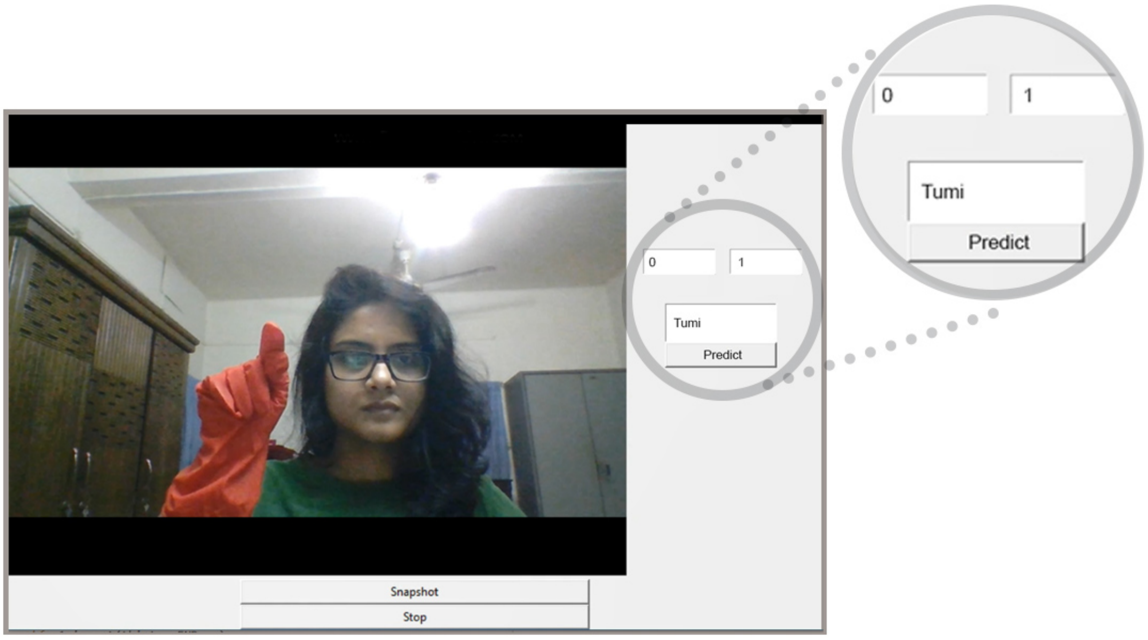
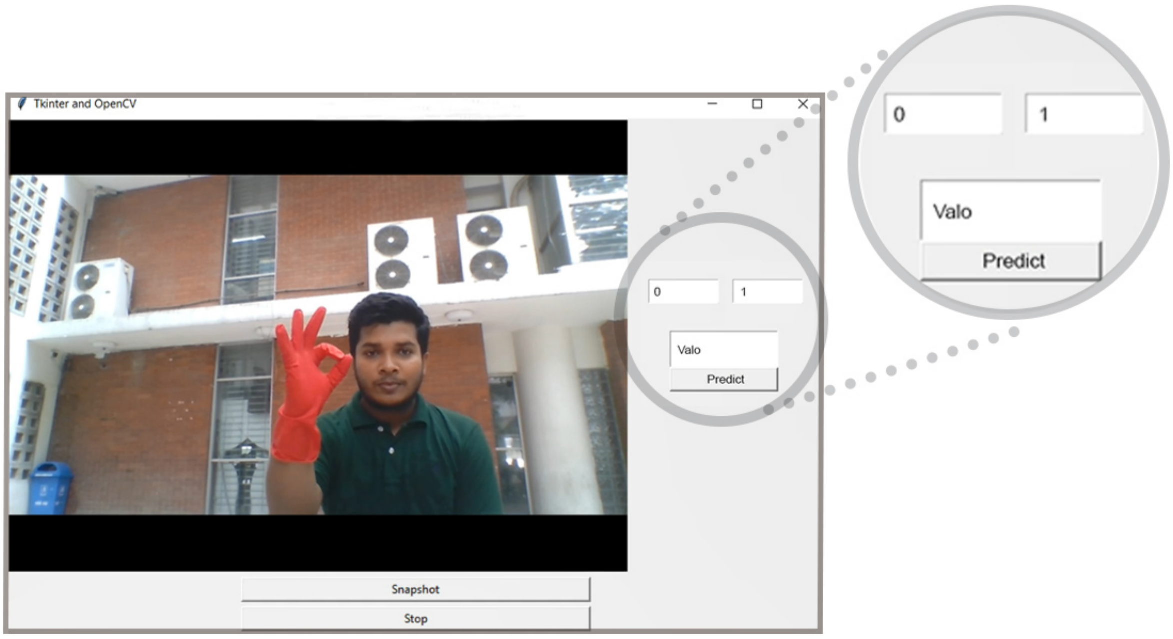


Figure 5.4: This figure represents a visualization of how frames spatial values are passed in LSTM model to train.

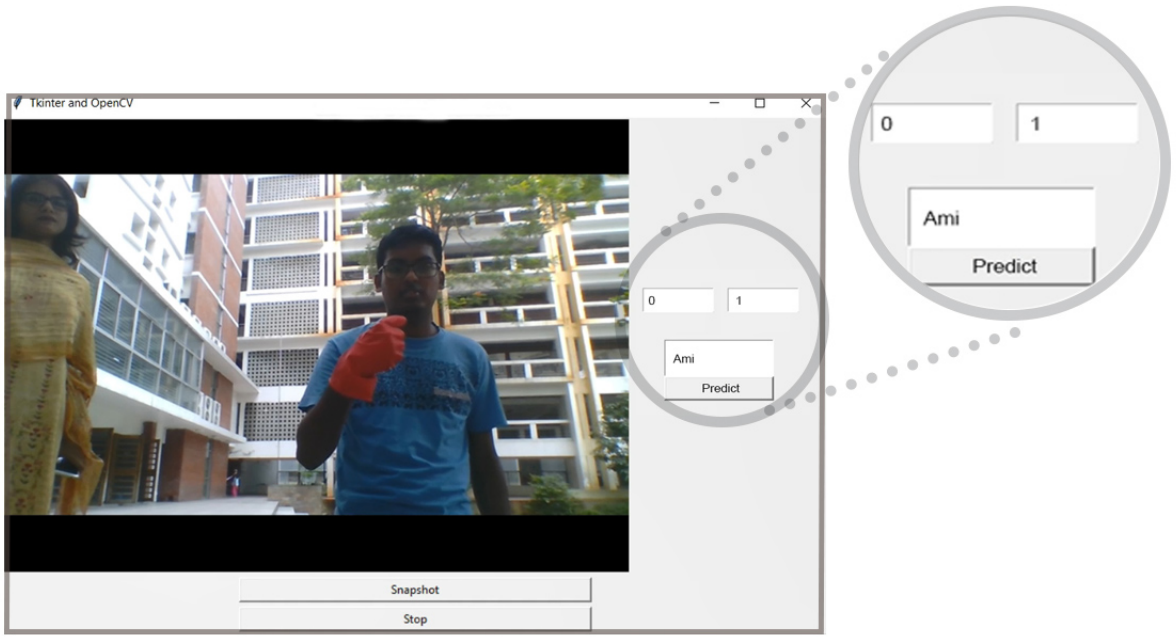


(a)

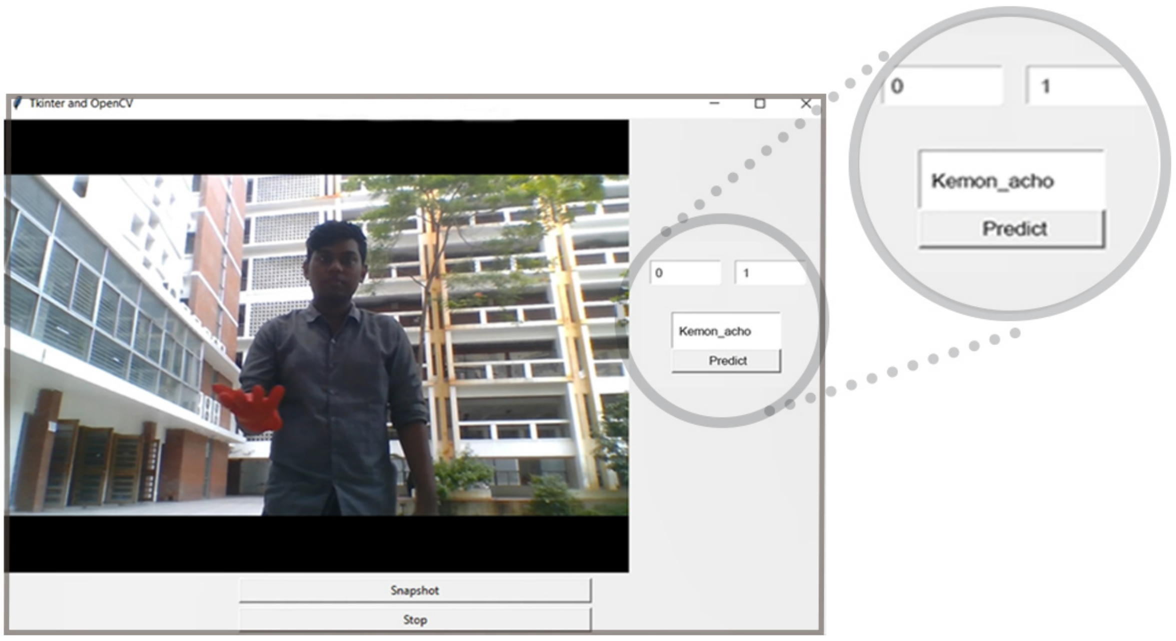


(b)

Figure 5.5: Some results of real time detection of word in different backgrounds by different subjects. The user first has to start recording a video of sign, by clicking the snapshot button. To finish recording stop button has to be pressed. After recording, the user has to give input of the start (t_1) and end (t_2) time from the recorded video, containing sign in that portion and then click predict button to see the results. In (a) start time is 0 second and end time is 1 second. These inputs are given by the user. The system has predicted 'Tumi' for that duration of video & in (b) 'valo' has been recognized similarly.



(a)



(b)

Figure 5.6: Continuation of Figure 5.5. Some results of real time detection of word in different backgrounds by different subjects. (a) 'ami' and (b) 'kemon_acho', these classes have been recognized by our system on a real-time recorded video in different backgrounds by different subject.

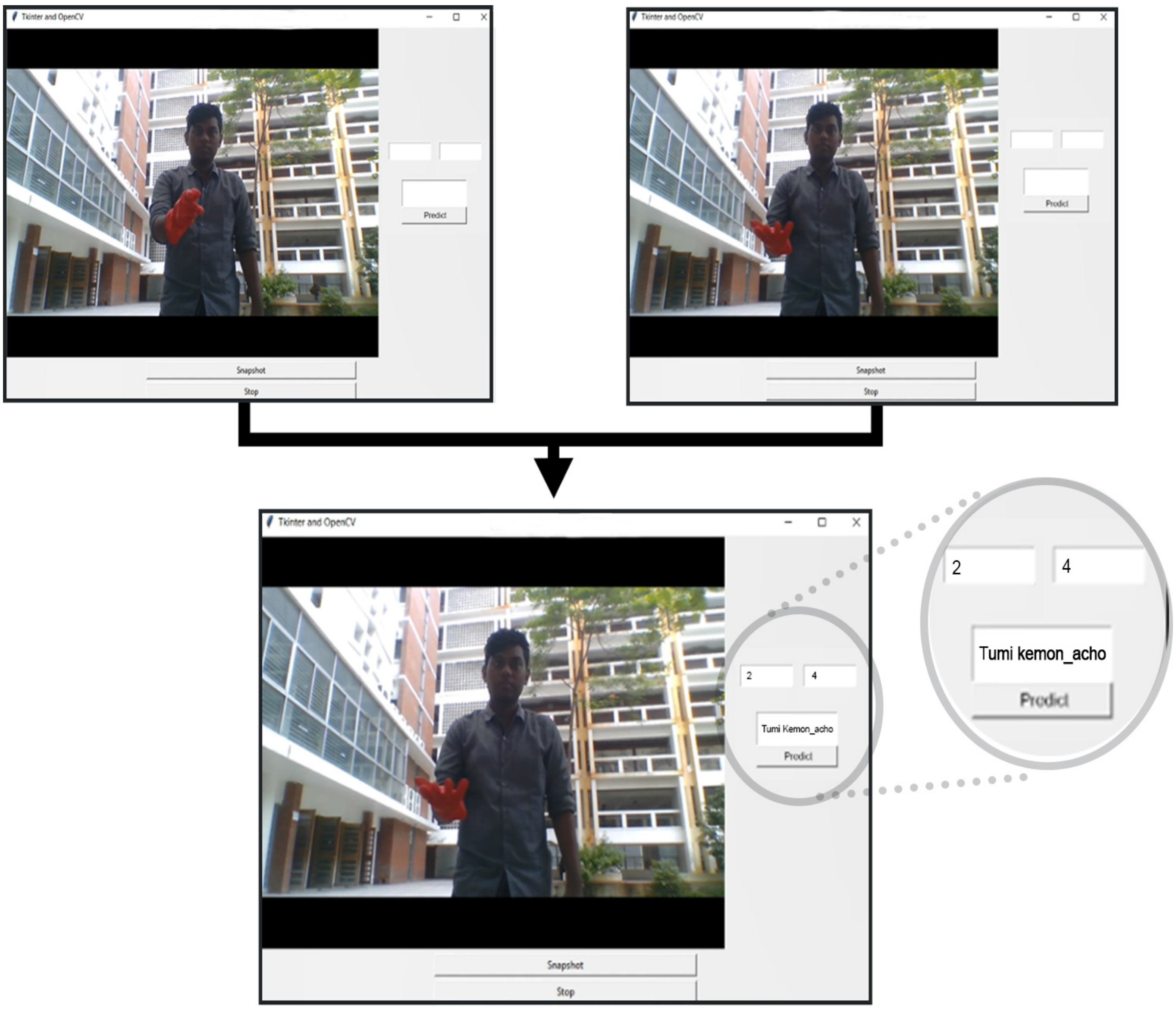


Figure 5.7: Continuation of Figure 5.5. In this output, system has recognized two gestures at a time. User has given input of $t_1 = 2$ second and $t_2 = 4$ second after recording the video in real-time. This duration of video contained two sequential gestures. And the system has recognized two gestures successfully. Output of two recognized class 'Tumi' and 'Kemon_acho' is displayed in the box by the system.

Chapter 6

Conclusion and Future Work

In this paper, we have developed a system that would recognize BdSL-Letters and BdSL-Words in real time.
We have created our own *BdSLImset* for our system. Our dataset has 10 different labeled sign letters. We collected about 200 pictures for each sign. Images of different BdSL signs from our *BdSLImset* dataset were trained by Faster R-CNN based model to solve the problem of sign language recognition. We obtained an average accuracy rate of 98.2 percent and recognition time was 90.03 milliseconds.
We have generated our own *BdSLVidset* for our system. We collected about 50 videos for each gesture where 5 subjects have participated. Each video was then represented by a sequence of predictions made by CNN model for each individual frames and this sequence of predictions was given as input to the RNN. The result is satisfactory for both models.

Different possible avenues of future exploration of our research are discussed below :

- The system has limitations while recognizing the letters, which have many similarities among their patterns. The problem might be overcome with more image data for those letters.

- The number of class in our video dataset, made for video recognition is very small, only for four signs. Dataset has to be increased for multiple classes. We also need to improve our system on word recognition for detecting the gestures at a much shorter time.

- Also, the image size is a factor, as data training requires a huge amount of time. Our research is still ongoing progress. As we will be needing huge amount of data for our work, for data collection process-

  — We need to conduct a number of workshops on Deaf School and we have already had a meeting with the principal of the school.

— We have also already arranged a meeting with the officials of the Centre for Disability & Development (CDD) to help us with our idea.

— We will be always need to keep communication with the signers to maintain the user-friendliness of our system.

- For better word recognition we need a much larger dataset, and for a feasible system for users, the system has to cope with multiple varying backgrounds and recognize the words without any added gloves. In the future, we have the plan to evaluate our model by genuine users to sort out its limitations and improve the system. This will also help us see how the system reacts to real-life situations and how clearly it can recognize the pattern and interpret effectively.
- We need to increase our dataset further for both BdSLImset and BdSLVidset, to make a complete dataset of all the letters and words.
- We plan to make a mobile system using YOLO. Our final goal is to merge both of our systems, and create a complete sign language application, which can both recognize all BdSL-Letters and BdSL-Words in real time.

References

[1] "Tensorflow models." https://github.com/tensorflow/models/tree/master/research/object_detection. Accessed:2019-06-3.

[2] "Masking R-CNN." https://www.learnopencv.com/. Accessed:2019-06-4.

[3] "YOLO: Real-Time object detection." https://pjreddie.com/darknet/yolo/. Accessed:2019-06-4.

[4] "Long Short-Term Memory (LSTM): kernel concept." https://medium.com/@kangeugine/. Accessed:2019-06-2.

[5] "Understanding lstm networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed:2019-06-2.

[6] "Long short term memory cells." https://machinelearners.net/2017/08/23/. Accessed:2019-06-3.

[7] M. Fayyaz, M. Hajizadeh Saffar, M. Sabokrou, M. Fathy, R. Klette, and F. Huang, "Stfcn: Spatio-temporal fcn for semantic video segmentation," 08 2016.

[8] S. Masood, A. Srivastava, H. Thuwal, and M. Ahmad, *Real-Time Sign Language Gesture (Word) Recognition from Video Sequences Using CNN and RNN*, pp. 623–632. 01 2018.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," vol. abs/1506.01497, 2015.

[10] M. B. Waldron and S. Kim, "Isolated asl sign recognition system for deaf persons," vol. 3, pp. 261–271, Sept 1995.

[11] A. Karami, B. Zanj, and A. K. Sarkaleh, "Persian sign language (psl) recognition using wavelet transform and neural networks," vol. 38, (Tarrytown, NY, USA), pp. 2661–2667, Pergamon Press, Inc., Mar. 2011.

[12] C. Wang, W. Gao, and Z. Xuan, "A real-time large vocabulary continuous recognition system for chinese sign language," in *Advances in Multimedia Information Processing —*

PCM 2001 (H.-Y. Shum, M. Liao, and S.-F. Chang, eds.), (Berlin, Heidelberg), pp. 150–157, Springer Berlin Heidelberg, 2001.

[13] O. B. Hoque, M. I. Jubair, M. S. Islam, A. 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)*, pp. 1–6, Dec 2018.

[14] F. Ronchetti, F. Quiroga, C. Estrebou, L. Lanzarini, and A. Rosete, "Lsa64: A dataset of argentinian sign language," 2016.

[15] F. Yasir, P. Prasad, A. Alsadoon, A. Elchouemi, and S. Sreedharan, "Bangla sign language recognition using convolutional neural network," in *Proceedings of the 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies*, (United States), pp. 49–53, IEEE, Institute of Electrical and Electronics Engineers, 4 2018.

[16] M. A. Rahaman, M. Jasim, M. H. Ali, and M. Hasanuzzaman, "A real-time appearance-based bengali alphabet and numeral signs recognition," vol. 4, Dhaka University, 2017.

[17] B. Chandra Karmokar, K. Alam, and M. Siddiquee, "Bangladeshi sign language recognition employing neural network ensemble," vol. 58, 10 2012.

[18] 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, Dec 2016.

[19] Y. Ji, S. Kim, and K.-B. Lee, "Sign language learning system with image sampling and convolutional neural network," pp. 371–375, 2017.

[20] 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)*, pp. 192–197, Dec 2014.

[21] M. A. Rahaman, M. Jasim, T. Zhang, M. H. Ali, and M. Hasanuzzaman, "Real-time bengali and chinese numeral signs recognition using contour matching," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1215–1220, Dec 2015.