

# A Comparative Study on Bengali Handwritten Character Recognition and Prediction using CNN

by

Muntaqa Abrar  
17101288  
Md Nazial Kadir  
17101100  
Tabassum Faruk  
17101493

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

Department of Computer Science and Engineering  
Brac University  
January 2021

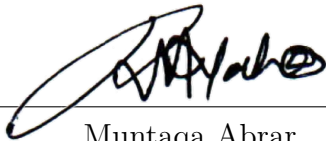
© 2021. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

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

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



---

Muntaqa Abrar  
17101288



---

Md Nazial Kadir  
17101100



---

Tabassum Faruk  
17101493

# Approval

The thesis titled “A Comparative Study on Bengali Handwritten Character Recognition and Prediction using CNN” submitted by

1. Muntaqa Abrar (17101288)
2. Md Nazial Kadir (17101100)
3. Tabassum Faruk (17101493)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 20, 2021.

## Examining Committee:

Supervisor:  
(Member)



---

Md. Saiful Islam  
Lecturer

Department of Computer Science and Engineering  
Brac University

Program Coordinator:  
(Member)



---

Md. Golam Rabiul Alam, PhD  
Associate Professor

Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

---

Mahbubul Alam Majumdar, PhD  
Professor

Department of Computer Science and Engineering  
Brac University

## **Ethics Statement**

As we had to collect primary data of handwriting samples from various volunteers from different institutes, we are committed to keep all the respondent's identities anonymous. This data will be used only for research purpose.

## Abstract

The transcription Bengali text to digital text is neither very efficient nor accurate. This proves to be a problem because most official work in Bangladesh is traditionally done in Bengali, on pen and paper hardcopy documents, which are difficult to transition to digital format. In our thesis, we attempted to solve this problem by improving the process of recognizing and extracting handwritten Bengali text to digital text. To aid us in our research, we have also collected an extensive data set consisting of approximately 25000 samples of around 90 Bengali characters each, including conjunct characters, to help us establish our findings. The main models we have implemented in our paper are- VGG-19, ResNet50, AlexNet, SqueezeNet. The highest training accuracy was 87% and was achieved from AlexNet, and least was 54% from VGG-19. The reliability of our model was validated by F1 score.

**Keywords:** Handwritten Character Recognition; Bengali Characters; Image Processing; Convolutional Neural Network; AlexNet; SqueezeNet

## **Acknowledgement**

First and foremost, praises and thanks to Allah for His blessings that enabled us to complete our research successfully despite the many obstacles. We would like to express our deepest and sincerest gratitude to our thesis supervisor, Md Saiful Islam, for his continuous support and invaluable guidance throughout the duration of our thesis. We would like to express our deepest and sincerest gratitude to Medlar Apparels Limited and Agrani Bank Limited, for their participation in the data collection process of our research work. Finally, we would like to thank our parents and peers, for their help and motivation, without which we would not have been able to complete this research work.

# Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Aim of Study . . . . .	2
1.4 Research Methodology . . . . .	2
1.5 Thesis Outline . . . . .	4
<b>2 Related Work</b>	<b>5</b>
<b>3 Data Collection and Processing</b>	<b>9</b>
3.1 Data Collection . . . . .	9
3.1.1 Primary Data . . . . .	9
3.1.2 Secondary Data . . . . .	11
3.1.3 Data Comparison . . . . .	12
3.2 Workflow . . . . .	12
3.3 Binarization and Inversion . . . . .	13
3.4 Bounding Boxes and Extraction . . . . .	15
3.5 Augmentation . . . . .	15
<b>4 CNN Model Implementations</b>	<b>17</b>
4.1 CNN and Machine Learning . . . . .	17
4.2 Supervised Learning . . . . .	17
4.3 CNN Models . . . . .	18
4.3.1 Learnable parameters . . . . .	18

4.3.2	Hyperparameters, optimizers and activation functions . . . . .	19
4.3.3	VGG-19 . . . . .	19
4.3.4	ResNet-50 . . . . .	20
4.3.5	AlexNet . . . . .	20
4.3.6	SqueezeNet . . . . .	21
<b>5</b>	<b>Results and Analysis</b>	<b>22</b>
5.1	Comparison of the Models . . . . .	22
5.1.1	AlexNet . . . . .	22
5.1.2	ResNet-50 . . . . .	23
5.1.3	SqueezeNet . . . . .	24
5.1.4	VGG-19 . . . . .	24
5.2	Accuracy and Loss Values of the Models . . . . .	25
5.3	Precision, Recall and F1 Accuracy . . . . .	26
5.4	Tuning Hyper-parameters . . . . .	27
5.4.1	Train:Validation Split . . . . .	27
5.4.2	Optimizer . . . . .	27
5.4.3	Input Size . . . . .	28
<b>6</b>	<b>Conclusion and Future Work</b>	<b>29</b>
	<b>Bibliography</b>	<b>31</b>
	<b>Appendix: Prepared Forms</b>	<b>32</b>



# List of Figures

1.1	The first Bengali consonant in the Bengali alphabet- Ba (in black), and its vowel diacritic forms (in red). . . . .	2
1.2	Our collected sample of Bengali alphabet- Ba . . . . .	3
3.1	Some of the sample data collected . . . . .	10
3.2	Sample data containing noisy input and ambiguous handwriting . . .	11
3.3	Same sample data as figure 3.2 after performing bounding box and ostu method binarization . . . . .	11
3.4	Workflow for Data Collection and Processing . . . . .	13
3.5	Result of local binarization (a) and global binarization (b) on a section of the data set . . . . .	14
3.6	Section of data set after binarization by otsu method followed by inversion . . . . .	15
3.7	(a) Initial Bounding Boxes (b) Final Bounding Boxes . . . . .	15
3.8	A final character cropped out of a data set form . . . . .	15
3.9	Some results of augmentation of a single character . . . . .	16
4.1	Supervised learning process [15] . . . . .	18
4.2	Architecture of VGG-19 . . . . .	20
4.3	Architecture of AlexNet Model . . . . .	21
4.4	Architecture of SqueezeNet Model . . . . .	21
5.1	Training accuracy and loss of AlexNet Model after 15 epochs . . . . .	23
5.2	Validation accuracy and loss of AlexNet Model after 15 epochs . . . . .	23
5.3	Training accuracy and loss of ResNet Model after 10 epochs . . . . .	23
5.4	Validation accuracy and loss of ResNet Model after 10 epochs . . . . .	24
5.5	Training accuracy and loss of SqueezeNet Model after 15 epochs . . . . .	24
5.6	Validation accuracy and loss of SqueezeNet Model after 15 epochs . . . . .	24
5.7	Training accuracy and loss of VGG-19 Model after 10 epochs . . . . .	25
5.8	Validation accuracy and loss of VGG-19 Model after 10 epochs . . . . .	25
5.9	Relative Training Accuracy Achieved by the Models . . . . .	25
5.10	Relative Training Loss Achieved by the Models . . . . .	26
5.11	Accuracies of AlexNet on 80:20 split (a) and 70:30 split (b) . . . . .	27
5.12	Variation in accuracy due to different input image sizes for AlexNet . . . . .	28

# List of Tables

3.1	A comparison of the primary and secondary data sets used in our thesis	12
5.1	A comparison of the models we have used . . . . .	22
5.2	Recall, Precision and F1 Accuracy of ResNet-50 and AlexNet . . . . .	26
5.3	Comparison of SGD and Adam optimizer on AlexNet . . . . .	27

# Chapter 1

## Introduction

### 1.1 Overview

Nowadays, the process of transcribing handwritten text to digital format has become very accessible due to great advancements in technology. It is now possible to instantly scan handwritten texts from documents, images, certificates and convert them to digital documents with great ease and accuracy for languages such as English and Mandarin. However, when transcribing Bengali handwriting, it yields many inaccurate results. This can be attributed to the fact that the Bengali alphabet has multiple characters that share similar curvature features. Hence, Bengali text transcription remains a vastly underdeveloped sector. Since Bengali is the official language of Bangladesh, it is used in most official work. However, since most of these documents are written by hand, they pose many problems such as storage, inefficiency, and reduced productivity etc. If these documents could be transcribed to digital text, then these problems could be easily resolved. Thus, in this paper, we have aimed to tackle the problem of scanning, identifying, and converting handwritten Bengali characters to digital format. We have also collected an extensive data set consisting of approximately 25668 samples of around 153 Bengali characters including 90 conjunct characters, to help us establish our findings. The main CNN models we have implemented in our paper are- VGG-19, ResNet-50, AlexNet, SqueezeNet.

### 1.2 Problem Statement

In recent times, there have been many great advancements in image processing and natural language processing. However, the digitization of handwritten text can be greatly improved. For more commonly-spoken languages, like English and Mandarin, there exist a few optical character recognition tools that are capable of extracting text from scanned documents, images etc. However, applying these tools to identify Bengali text has been problematic. This can be attributed to the structure of the Bengali alphabet, which has multiple characters that are very similar-looking, with nearly identical slants and curves. To that extent, much research has been conducted, as discussed in the literature review section. Overall, it has been observed that the process of detecting Bengali handwriting lacks accuracy, and is in fact unable to differentiate between similar-looking letters. Bengali is spoken by over 210 million people all over the world, and is the state language of Bangladesh, as

well as one of the official languages of India [8]. In Bangladesh, most official work is conducted in Bengali, and is traditionally written by hand. Bengali is used in conducting all levels of national examinations. It is also used in documenting important official work, including certificates, government policies, budgets etc. Over the years, these documents pile up and take up lots of space. Not only that, such paper-based storage is susceptible to wear and tear, and can be easily damaged by flood, fire etc. Storage and transport of hardcopies are troublesome and take up large amounts of space.

To resolve the problem of storage space, it is possible to instead scan and store these documents. However, that still does not resolve the problem of having to manually browse through large quantities of data to find relevant information.

A more prominent solution to the issue of manual storage of Bengali documents is to convert and store the documents as digital text. Digitizing these documents would forgo the need for physical space and also greatly reduce any risk of damage. It would also make it very easy to search for documents using keywords and also secure the papers from being lost. Hence our contribution may prove useful to government and non-government organizations and institutions in many ways.

### 1.3 Aim of Study

The aim of our study is the detection and recognition of Bengali characters from handwritten text, and converting them to digital format. In our thesis we have discussed our data set of 25668 samples of 153 Bengali characters each- including 11 vowels and 39 consonants, and a combination of 90 selected conjunct characters. To collect this data, we generated 3 distinct forms containing various combinations of the required characters, and collected handwriting samples from volunteers. The sample data was then sorted, processed and analyzed as per our study requirements. Using these methods, we implemented models on our data, and found accuracy rates and made comparative studies.

### 1.4 Research Methodology

Our target is to detect Bengali characters from handwritten Bengali text, and convert them to digital text. With this objective in mind, we generated 3 forms with random Bengali characters and collected handwriting samples from volunteers.

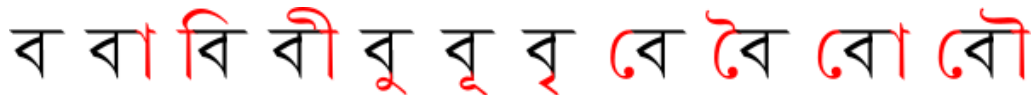


Figure 1.1: The first Bengali consonant in the Bengali alphabet- Ba (in black), and its vowel diacritic forms (in red).

Standard Bengali alphabet consists of 11 vowels and 39 consonants. Among these, each consonant can have one or more conjunct diacritic forms. In common, day-to-day Bengali, words are composed of consonants appended with vowel diacritics

or consonant diacritics. This conjunct form of characters is known as “Kar” in Bengali, and is an essential component of Bengali language. Figure 1.1 outlines the vowel diacritic forms or “Kar” forms of the Bengali consonant, “Ba” (ব). For our research purpose, we have decided to focus on such simple diacritic forms of Bengali characters.



Figure 1.2: Our collected sample of Bengali alphabet- Ba

As it stands, the type of data we needed was not readily available, as most existing Bengali data sets, like BanglaLekha-Isolated [3] contain only isolated Bengali characters and numerals, and not diacritic forms of characters. The data was collected from various places and from people of all ages- from school children to grown adults, from schools and from offices, so the handwriting samples collected contain a significant variation among them. After successful completion of data collection, we sorted through the collected data and performed various image processing techniques on the data, for example, for extracting the text, we implemented the Bounding Box algorithm on the forms. Then we implemented several deep convolutional network models, combining old fashioned models such as VGG-19, and then using more contemporary and advanced models such as ResNet-50, AlexNet, and SqueezeNet on our data set. To compare results and check the accuracies of our models, we utilized recall and F1 score.

## 1.5 Thesis Outline

This thesis report is a research which aims to find the best model for detection and recognition of Bengali handwritten characters. Our aim has been to collect a large data set, process them and perform character recognition on the collected data using various state of the art CNN models.

Additionally, we aim to find the best model which gives highest accuracy upon the collected data. The overall report focuses on the steps that have been followed in conducting this research. Firstly, the introduction (Chapter 1) addresses the motivation behind conducting this research, the problem statement, the aim of our study, a brief summary of our research methodology and a complete outline of our thesis paper.

Then, in the literature review section (Chapter 2) there is a discussion of existing papers and research that have been conducted on this particular issue. We have studied these papers extensively and relied on the knowledge and ideas in them to improve our understanding of our problem. Upon conducting this background study, we were able to find the shortcomings and lackings in existing research. This has helped us to design a more expansive data set, and devise more accurate solutions to the existing problem.

In the data collection section (Chapter 3), we have explained the need for a primary data set and why we chose to collect primary data instead of using existing, secondary data. We also elaborated on the contents of our data set, and any special features of our collected data. Then we described the process of our data collection in a workflow diagram. Afterwards, we explained our data pre-processing techniques- Bounding Box, Extraction, Augmentation and Binarization.

Then in the CNN model implementations section (Chapter 4) we have discussed all the models that we have studied in our research. We started with a brief discussion of CNN and Machine Learning, Supervised Learning, learnable parameters, hyperparameters, optimizers, and activation functions. Then we discussed the CNN Models that we used in our study- VGG-19, ResNet-50, AlexNet, and SqueezeNet.

In the results and analysis section (Chapter 5) we compared the accuracy and loss of the models we implemented on our data. Then, we discussed precision, recall and F1 accuracy, and finally established our overall findings by discussing the train:validation split, and the effects of optimizer and input size on our data.

Finally, in the conclusion and Future Work section (Chapter 6) we drew a conclusion to our research paper and discussed our future plans regarding our research.

# Chapter 2

## Related Work

Much research has been done and developed over the past few years on Bengali language. We have found various research papers on Bengali language regarding OCR, sex detection from handwriting, etc.

Saha et al.[14] made it possible to detect sex from Bengali handwriting. The data set they used was BanglaLekha-Isolated[3]. This multi labeled data set consists of 166,060 gray scaled images. CNN was used to train their data set. The augmentation for training data was done artificially. To increase data, they resized and rotated all the images. After removing the possibility of data dependency from all the labels, Frobenius Normalization was used to regularize the loss of function. . Adam optimization algorithm made the optimization faster as well as combined the idea of gradient descent along with Momentum and Root Mean Square prop. The embedded data set was fed into the RNN and sex was the target label for each image. The researchers embedded the RNN to what type of alphabet the image represents. Long Short Term Memory (LSTM) was used to make the network remember long term dependencies. The data set was divided into 3 parts- training, cross-validating and testing. VGG-19, ResNet-125, GoogleNet, FractalNet, and DammNet were also used for training the data set. In terms of training time, parameters and classification accuracy, DammNet “in press” performs the best. This DCNN model decreases the complexity of learned function. Therefore, it is efficient in both time and memory. They achieved an accuracy of 91.85% on their RNN model. Furthermore, they got 89.37% accuracy on the data set (100 male and 100 female) they had collected on their own.

Similarly, BanglaLekha-Isolated data set was also used for training and testing by Hakim et al.[10]. Their research focused on recognizing Bengali handwritten numbers and basic characters by DCNN. They have prepared a 9 layered sequential CNN model to identify 60 Bengali characters. Their proposed model is a simpler version of VGG-16. The 60 characters include 10 Bengali numerals and 50 Bengali basic alphabets. For cross-validation another 6000 samples were collected. Improper samples were removed from the data set such as characters not having “mattrā.” The Otsu method was applied for binarization and Gaussian filter to reduce noise. Sample images were cut by the grid line and sorted into folders. Bounding box method was used to extract only the character part from the image. As CNN accepts only square images, samples were resized and padding was done. After

that, the images were labelled according to their class. To increase the size of the data set, data augmentation was done. They achieved 99.44% and 95.16% accuracy rate on BanglaLekha-Isolated data set and their collected data set respectively for 60 characters. Additionally, they achieved 99.6% and 99.82% accuracy rate on BanglaLekha-Isolated data set and their collected data set respectively for 10 numerals.

Chowdhury et al.[6] research was to recognize Bengali handwritten characters and convert them to editable text. This is similar to our work. Their data set contained 2000 samples for each Bengali character. They used MATLAB as their platform. For the training data set, a neural network was built with the help of neural network toolbox from MATLAB. After removing ambiguous samples, 700 samples per letter were taken. Then data were binarized and resized. The training was accomplished by the built-in feature of MATLAB. Image processing was done by MATLAB tools as well. The accuracy of the training data set ranged between 88% to 95% per character. They took images as input having the extension .jpg or .png. The output was shown on a text file.

Bhowmik et al.[2] conducted their research on character recognition depending on stroke features. They have used MLP classifiers for their work. They have extracted 10 certain features from each character. The training data set contains 350 and the test data set contains 90 sample images for each Bengali character. Firstly, the gray scaled images are median filtered and then binarization is done. Secondly, the digital curves representing the vertical and horizontal strokes are detected and then stored. Finally, extraction was done from feature strokes. Their MLP model gave the best result when the number of hidden layers was approximately 50% of their input layer size (feature vector size). An image has at most 10 feature vectors. For the final classification purpose, feature vector of 100 length was fed into the MLP classifier.

Another research has been conducted by Gupta et al. [9] which focused on offline English handwritten character recognition. For their research, they have used ANN, SVM, RBF, and MLP. Heuristic segmentation algorithm was applied to detect valid segmentations among the alphabets. For the segmentation purpose, they created their own database containing 260 data. For training and testing, The Chars74K data set was used. The classifiers were tested in MATLAB by using the neural network toolbox. MLP classifier did not give a satisfactory result on validation and testing data set. RBF comparatively showed better results on validation data set but suffers from overlearning. SVM performed the best on the test data.

Furthermore, Roy et al.[13] focused on recognition on Bengali language online and by taking input in real time. They have considered 59 basic strokes. In this paper, stroke indicates the pen points between a pen up and down. They have used neural networks for their research purpose. When they feed a segmented portion to their neural network, it falls under in one of the 59 classes. But the success rate for such cases was not satisfactory. Hence, they planned to use a two phased neural networks targeting to increase the success rate by decreasing the number of output classes. They have used a digital notepad for taking input as the recognition was done in



real time. For their research, they have taken the pen positions and pen pressure at particular intervals. They considered 233 features for character recognition. The features consist of - structural, point based and directional features. Basic strokes were divided into five classes- special, valid character class, maybe valid character class, modifier character class, not character class. After pre-processing the scanned data, it goes through feature extraction. Then the strokes go through the two phased neural network. Constituent strokes of the input character are detected on the basis of its corresponding class category on the first phase of the neural network. The actual character id of the strokes is identified on the second phase of the neural network.

Chowdhury et al.[5] proposed a system where handwritten Bengali characters can be detected and converted to digital format. They hosted their model on a web-server as it was easier for them to interact with the model. For their research, they have used BanglaLekha-Isolated data set. They have achieved an accuracy rate of 91.81% by using CNN. Afterwards, they increased the number of total images to 200,000 with the help of data augmentation and got the accuracy rate of 95.25% on the test data set. Their aim for this research was to create such a model which can not only train on a large amount of data but also classify new images in real time. They have two convolutional layers containing 32 filters having the kernel size 5x5. Both have the activation function ReLu which is followed by the Max pooling layer. Three fully connected layers containing 1024 nodes can be found afterwards. All of them have the ReLu activation function. First of these three layers have a Dropout layer and the third layer have the output layer. The activation function used by the output layer is Softmax and the optimization function for this research was Stochastic Gradient Descent. Validation loss for the augmented data set was much less than the initial data set.

Alom et al.[1] conducted research on recognition of Bengali handwritten characters using State-of-the-Art DCNN. They compared their result with other most used recognition models and found that their approach shows much better results. Their research included Bengali alphabets, numerals and special characters. The DCNN models that they have used were VGG-16, All-Conv, NiN, ResNet, FractalNet and DenseNet. For testing and training, they used The Alphabet-50 data set. The evaluation of their models were done on three data sets from CMATERd. In case of testing accuracy, DenseNet outperformed all other DCNN whereas All-Conv gave the worst result. In terms of validation accuracy and convergence with other models, DenseNet and FractalNet performed the best among all the other DCNN models used for this research. The researchers have achieved accuracy rates of 98.13%, 98.31%, 98.18% for Bengali handwritten numerals, characters and special characters respectively by using DenseNet.

Our work is similar to Chowdhury et al.[6] research. But they have used MATLAB and its built-in toolbox for their work. On the other hand, we have used python's OpenCV library. We have used VGG-19 and ResNet-50 to build our neural networks whereas they have used neural network toolbox to build theirs. Saha et al.[14] for training their data set, they have used multiple CNN. In their work, DammNet produced the best result whereas among VGG-19 and ResNet-50, ResNet

produced the best result in ours and VGG-19 gave the worst accuracy. Hakim et al.[10] designed a simpler version of VGG-16. We on the other hand we have used VGG-19 and ResNet-50. In Roy & Sen's[13] paper, they have taken input in real time. They categorized the characters depending on the strokes. We took previously scanned documents as input and pre-processed the inputs through various steps using different methods.

Moreover, we have used the deep neural network model SqueezeNet and the convolutional neural network model AlexNet for our research purpose. These models have not been used before in similar work like ours to our knowledge. In our case, AlexNet gave us the best result in terms of both training and validation accuracy.

# Chapter 3

## Data Collection and Processing

### 3.1 Data Collection

#### 3.1.1 Primary Data

Bengali handwritten data sets are not very common or readily available. There exists a few large data sets such as BanglaLekha-Isolated [3] but they don't contain diacritic forms of Bengali characters which are crucial to our research. Hence, we have resorted to collecting data locally from Bangladesh, and have prepared 3 separate forms for collecting handwritten data anonymously. Due to restrictions and closures imposed on Bangladeshi educational institutions and offices alike for the past few months, our data collection process has been greatly hindered. Despite many obstacles, we have maintained proper precautions, and have been able to collect a moderate amount of data, from people of various ages and occupations. We plan to expand and enrich our data set further in the near future.

Bengali language is a rich language with 50 characters in the alphabet, of which there are 11 vowels and 39 consonants. For our thesis, we have created a data set of Bengali characters that currently contains about 25668 samples. The data set contains all 11 vowels and 35 consonant graphemes in isolated form. The rest of the last 4 consonant graphemes are adjoined to one base character (ব) or “ba”. Additionally, there are 13 of the previously discussed “kar” or vowel diacritics appended to the aforementioned base character. The data set also contains a total of 90 conjunct compound Bengali characters or “Juktakkhor”, which we have chosen not to use for our current thesis purpose.

Initially, the Bengali handwriting data was collected from approximately 300 volunteers by using forms printed on standard A4 paper, as shown in the figure 3.1. The form contained square grids of sample characters and blank boxes adjacent to them, where the volunteer filled out the characters using pen in their own handwriting. These paper forms were then scanned with a dimension of 2400x3500 pixels at 200dpi, and then sorted and separated according to the form numbers.

In the first stage, the data was extracted from these scanned forms. Due to time and resource constraints, we had to resort to manually cut out characters from the scanned images and collect the required data. Later on, we extracted the data

অ	অ	আ	আ	ই	ই	ঈ	ঈ
উ	উ	ঊ	ঊ	ঋ	ঋ	এ	এ
ঐ	ঐ	ও	ও	ঔ	ঔ	ক	ক
খ	খ	গ	গ	ঘ	ঘ	ঙ	ঙ
চ	চ	ছ	ছ	জ	জ	ঝ	ঝ
ঞ	ঞ	ট	ট	ঠ	ঠ	ড	ড
ঢ	ঢ	ণ	ণ	ত	ত	থ	থ
দ	দ	ধ	ধ	ন	ন	প	প
ফ	ফ	ব	ব	ভ	ভ	ম	ম
য	য	র	র	ল	ল	শ	শ
ষ	ষ	স	স	হ	হ	ড়	ড়
ঢ়	ঢ়	য়	য়	ব্	ব্	বৎ	বৎ

Figure 3.1: Some of the sample data collected



significantly less than our target of 1000 volunteers. Since the amount of data we collected in our primary stage was insufficient for our research purposes, we had to resort to utilising a second source of data, which is BanglaLekha Isolated.

BanglaLekha Isolated is a rich data set of size 166015. It contains 84 different characters among which there are 50 Bengali basic or simple characters, 10 numerals or numbers, and 24 compound characters. The data in this data set has been collected similarly to ours- as in, the pen-and-paper raw data had been collected on printed forms, and later scanned at 600dpi.

The collected data had then been pre-processed, the handwritten characters extracted and verified manually. The background and text had also been inverted to aid in the machine learning processes, and a median filter applied for noise reduction purposes. Additionally, the data samples are labelled according to location, institution of data collection, sex, age, and date of form entry, as well as the form serial number. The label of each form was 22 characters long, and due to the vast amount of data present, the data can be effortlessly processed to retrieve required information.

Therefore, we also used BanglaLekha Isolated to run and test the accuracy of our models and processes, which will be discussed in detail later.

### 3.1.3 Data Comparison

Since we utilised two separate sources of data for our research, we have made a comparison as shown in table which shows the different features and characteristics of our data sets.

Dataset	Number of total Samples	Number of total Simple characters	Number of total Conjunct/Compound characters	Number of total Diacritic form	Number of total Numerals
BanglaLekha Isolated	166015	98950	47407	0	19748
Primary Data collected by us	25668	14628	8280	2760	0

Table 3.1: A comparison of the primary and secondary data sets used in our thesis

## 3.2 Workflow

After acquiring the data set collected over several weeks, the bulk of the work methodology moved on to processing the data to ready it for the convolutional neural networks. This involved reducing data size to increase processing speeds, simplifying

the information to eliminate redundancies and augmenting the characters to expand the scope of our data set.

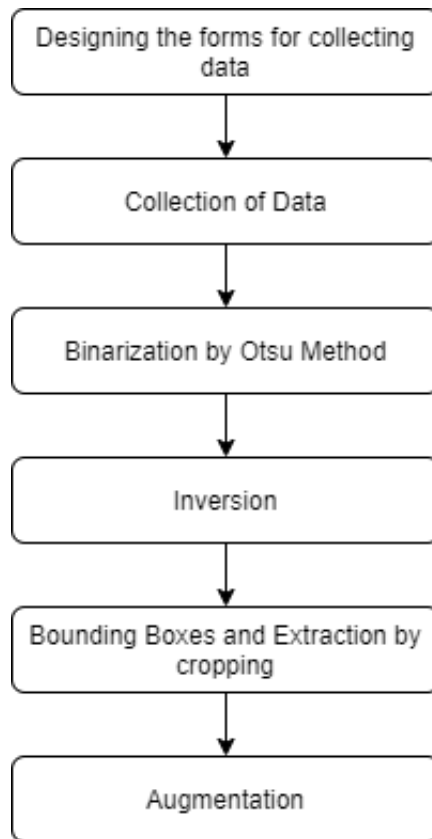


Figure 3.4: Workflow for Data Collection and Processing

### 3.3 Binarization and Inversion

The first steps to prepare the scanned forms involved reducing the resolution of the images obtained. On average, the scanned forms were of resolution 2400x3500. After testing out different resizing ratios, we concluded that half the original resolution (1200x1750) was a proper compromise between reduction of file sizes and loss of resolution clarity.

The first step of processing individual data is binarization of the forms. Binarization can only be conducted on single channel images, so the original forms were converted to gray scale to eliminate the 3 channel RGB format.

Binarization is a method that converts gray scale images to black and white images, assigning values 0 (for black) or 1 (for white) to each pixel [12]. A gray scale image contains an intensity value of between 0-256 for each pixel. This makes each image hold a great amount of information and hence analysis of such images can be computationally expensive. The data set we process in this paper consists of mere handwritten characters where intensity of color values is redundant as color is not an identification feature for the characters. Binarization of our data set not only makes the processing of the data faster, but also makes recognition far more

accurate as the algorithm must identify only one of two values for each pixel.

Binarization assigns a threshold value for the split between black and white. By comparing the intensity of grey level for each pixel against the threshold, the algorithm decides if the pixel should be assigned as black or white. For example, if the threshold value is set at 180, grey values of between 180-256 would be converted to white, while those below will be converted to black.

Broadly, the process can be categorised into two methods - global binarization and local binarization. In local binarization, the process selects a section of the image, calculates an average threshold value for that particular window of image and converts the pixels in only that section into black and white before moving onto a different section. This works well for multi-tonal images as discrepancies between layers and grey levels can preserve detail in the image. In global binarization, a global threshold value is set for the entire image and all pixels are converted based on the same scale. We tested our data set using a local binarization method and otsu binarization, a global process.

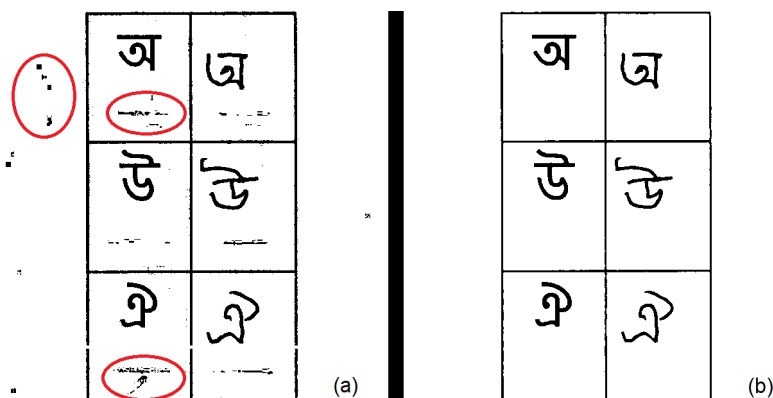


Figure 3.5: Result of local binarization (a) and global binarization (b) on a section of the data set

Figure 3.5 shows the results of local and otsu binarization. Local binarization produced very noisy images as the more accurate algorithm identified slight grey levels that exist on our scanned forms. This is undesirable as that high a level of detail is redundant for our simple character forms in the data set. Otsu binarization produced much cleaner images with little to no noise as the threshold was much higher and ignored slight grey markings on the paper. So for our methodology, we used Otsu binarization to convert the scanned forms.

In a black and white image, the white (1) pixels are the pixels with a value, while black pixels (0) are identified as having no value. Since after binarization, the characters were in black on a white background, each image contained all its values within the background, which is much greater in area than the character itself. To reduce the information size of the images, they were inverted, letting the computer ignore the background entirely as it identifies the character displayed in white (Figure 3.6).



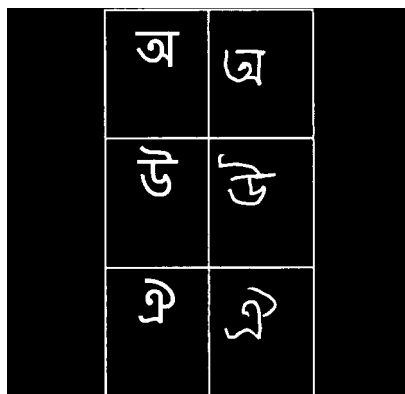


Figure 3.6: Section of data set after binarization by otsu method followed by inversion

### 3.4 Bounding Boxes and Extraction

In order to extract only the character areas out of the forms, a bounding box method was used. The initial result of bounding box predictions produced boxes that often included two closely positioned characters in a single box, as shown in Figure 3.7a.



Figure 3.7: (a) Initial Bounding Boxes (b) Final Bounding Boxes

To mitigate this issue, the program was modified to allow boxes under a threshold area value to be predicted and drawn after which we acquired box areas consisting of single characters.

The bounds also produced the coordinates of the vertices of each bound which were used in a program loop to crop out the individual characters from the forms. The average resolution of individual character images obtained is 45x45 pixels.



Figure 3.8: A final character cropped out of a data set form

### 3.5 Augmentation

The COVID-19 pandemic and the subsequent restrictions prevented us from acquiring the target size of data sets we initially hoped to collect. By running the individual characters through reshaping programs, the separate characters were augmented to

create slight variations such as tilts, resizing and shearing. This allowed us to produce more characters to expand upon the directly acquired data and form a much larger data set.

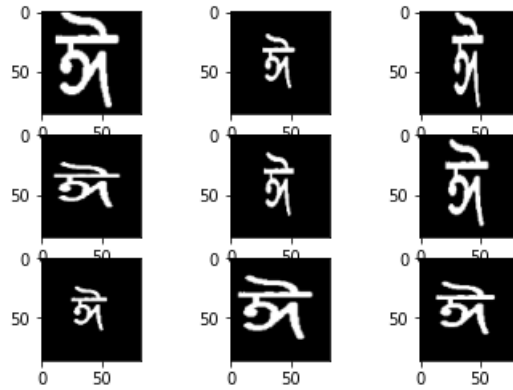


Figure 3.9: Some results of augmentation of a single character

# Chapter 4

## CNN Model Implementations

### 4.1 CNN and Machine Learning

Computer vision is a comprehensive domain of artificial intelligence that aims to enable computers and machines to interpret the world in a visual manner similar to humans. With the help of machine learning the field is designed to analyze data such as images, video and audio to recognise patterns for identification and classification of everyday objects, through the use of many classes of algorithms. With high processing speeds and efficient learning capabilities through training, convolutional neural networks (CNN) are among the most popular computer vision algorithms to analyze two dimensional imagery.

Convolutional Neural Networks are combinations of 3 primary types of layers - convolutional layers, which assign weights to and extract features from the input data, pooling layers that reduce the size of extracted features to increase processing speeds, and fully connected layers to use the features to classify the data.

### 4.2 Supervised Learning

Learning and categorizing data in CNN models can be done in a number of ways, the most prominent being supervised and unsupervised learning. Unsupervised learning requires the model to extract similar features from a training set of unlabelled data and hence form clusters of the data based on the proportions of similar features. In our models, the processed data have been manually separated into training and testing groups and the training set has been organised into different directories for each character to be identified. Hence, the paper used a supervised method of learning. The algorithms identify the separate directories as being separate classes of objects and do not require predictive clustering of the data to assume type labels. The test set consists of data which did not exist in the labelled training data, hence the accuracy of the models can be tested using data that was not learned from.

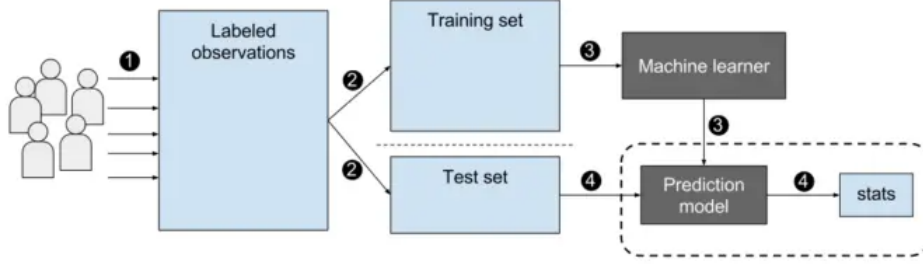


Figure 4.1: Supervised learning process [15]

## 4.3 CNN Models

For our analysis, a sample of our collected data-set was trained and tested against CNN models based on VGG-19, ResNet-50, AlexNet, SqueezeNet. The hyper-parameters of the models needed sufficient fine-tuning to increase accuracy. The input channel was modified to ‘gray scale’ from the default ‘RGB’ as computation is significantly faster when images are analyzed in gray-scale. The model was trained on a batch size of 30 and was also validated on a batch size of 30. Batch sizes were also tested at values 10, 32 and 64.

### 4.3.1 Learnable parameters

In a neural network, learnable parameters are parameters that the network learns during the training period, such as weights or biases [17]. Initially, it is initialized as arbitrary values, which are later updated as the network is trained using large data samples, during the back-propagation process. The algorithm and the optimization functions change the values of the parameters. For a simple connected network with the following layers: input ( $m$ ), hidden and output ( $n$ ) layers, the total number of learnable parameters is the sum of learnable parameters in each layer, except the initial input layer. That is because the input layer is simply the input shape, hence there is nothing for the model to learn here. Therefore, for every other layer, the number of learnable parameters =  $m * n + bias$ , where bias is the number of nodes in that layer. And the total number of learnable parameters of the model is

$$\sum m * n + bias \quad (4.1)$$

Moving on to convolutional layers, the calculation for learnable parameters involves calculating the number of parameters in these hidden convolutional layers as well as the fully connected layers. Similar to the previous formula, the input layers and the pool layers do not have any learnable parameters.

For a convolutional layer with width  $m$ , height  $n$ , previous layer’s filters  $d$ , and  $k$  filters in current layer, the number of parameters in that layer = ((shape of width of filter \* shape of height of filter \* number of filters in the previous layer + 1) \* number of filters in the current layer). The “+1” is added for the bias terms in each filter. This can be simplified to:

$$((m * n * d) + 1) * k \quad (4.2)$$

For a fully connected layer, with current layer neurons ( $c$ ), and previous layer neurons ( $p$ ), the number of parameters in each layer = (current layer neurons \* previous layer neurons) + 1 \* current layer neurons). Simplified, the formula is

$$(c * p) + 1 * c. \tag{4.3}$$

Finally, the total number of parameters in the CNN is the summation of all the parameters in all the layers, as calculated using equation (3).

### 4.3.2 Hyperparameters, optimizers and activation functions

The adjustable parameters of a model, that are fine tuned to obtain optimal results from the model, are called hyperparameters. The changing of hyperparameters can greatly affect the performance of a model and its learning. Some common hyperparameters are learning rates, number of epochs, and batch sizes.

Whether a network will be activated and produce an output or not, is determined by the activation function. There are many activation functions, but for the purpose of our research, we have used a combination of relu and sigmoid functions, with the sigmoid function being exclusively used in the output layer in VGG-19.

Learning rate determines the speed of the model's adaptation to the dataset [4], and is generally set between 0.01 and 0.0001. When the value of learning rate is too low, it can require more learning epochs, and can even take too long to reach a solution. In contrast, when the value of learning rate is too high, it may require less epochs to reach a solution. However, the solution reached may not be the best solution since the model may have converged by skipping too many steps. Hence, for our research purposes, after much trial and error, we have kept the learning rates at 0.01 for VGG-19 and SqueezeNet, and 0.1 for ResNet-50 and AlexNet.

Neural networks generally learn by implementing the gradient descent algorithm. Nowadays, more advanced algorithms have been developed which work better with certain models. Among them are two algorithms: Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam) [16], both of which are variants of the gradient descent algorithm. SGD is suitable for model implementations with a slow learning rate, as it selects subsets of data randomly from the entire dataset, and performs computation on these subsets. On the other hand, Adam is an adaptive algorithm that uses approximations of the gradients to tune the learning rates to get the optimum learning rate of the model. Both of these optimizers work very well with deep neural networks. For our thesis, we have used SGD in VGG-19 and ResNet-50, and Adam optimizer in SqueezeNet and AlexNet.

### 4.3.3 VGG-19

Hakim et al [10] used a model based on VGG-16 with 6 convolution layers and 2 fully connected layers. Our model based on VGG-19 is much deeper consisting of 19 layers - 16 convolution layers, 3 Fully connected layers consisting of 5 MaxPool layers and 1 SoftMax layer. All Convolution networks are followed by a Maxpool layer. At the end are the 3 Fully Connected layers followed by the Softmax layer.



the model. Our model based on AlexNet performed the best among the four tested models.

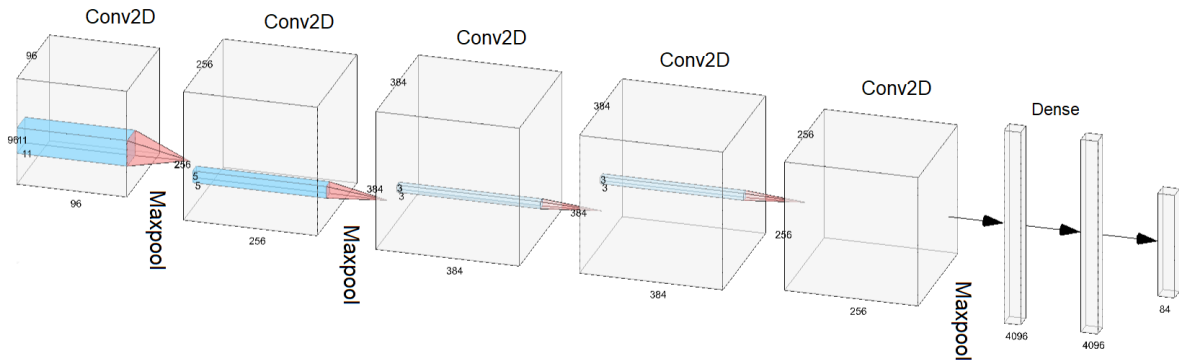


Figure 4.3: Architecture of AlexNet Model

### 4.3.6 SqueezeNet

With the conception of SqueezeNet, the algorithm was designed to challenge the efficiency and capability of AlexNet [11]. With almost 50 times less parameters, the deep neural network managed to achieve the high level of accuracy that AlexNet achieved at a greatly reduced size. Our model based on SqueezeNet has just over 700,000 parameters, using 10 convolution layers, 3 Maxpool layers and 1 Fully connected layer and also incorporates skip connections.

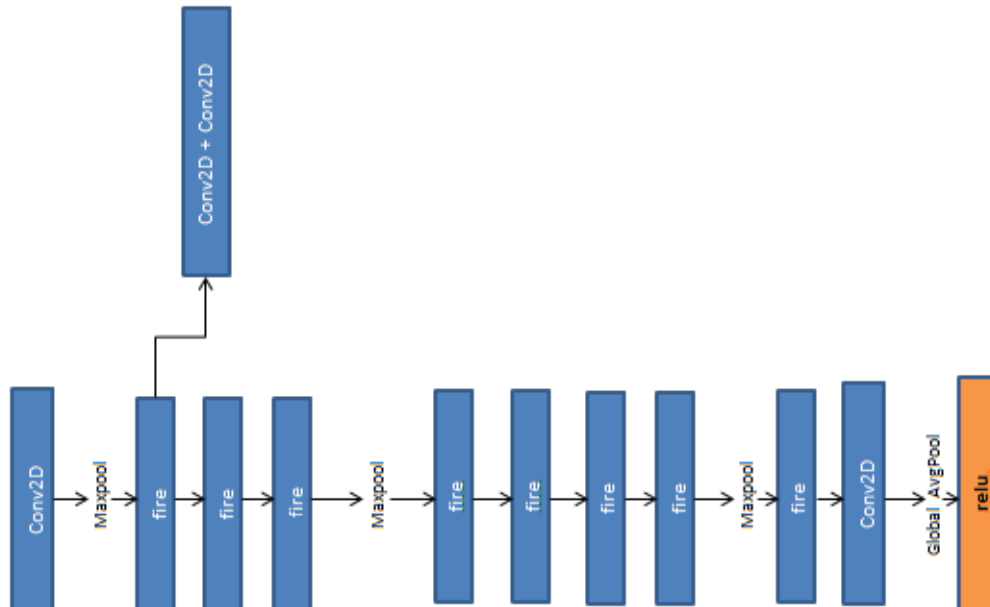


Figure 4.4: Architecture of SqueezeNet Model

# Chapter 5

## Results and Analysis

We implemented the neural network models on our labelled data set using varying hyper-parameters to optimize the performance of the individual models. The greatest challenge faced by our team to compute the performance of the chosen models was our lack of access to a system with sufficient computational power due to the lockdown imposed by the ongoing pandemic. High hyper-parameter values which could lead to larger values of accuracy were unattainable by the household computers available to us for use.

Model	Number of parameters	Number of epochs	Train-Validation Split	Training Accuracy	Validation Accuracy
VGG-19	39,184,404	10	70:30	54%	37%
ResNet50	26,340,308	15	70:30	78%	62%
AlexNet	25,050,260	15	70:30	87%	72%
SqueezeNet	769,108	15	70:30	63%	49%

Table 5.1: A comparison of the models we have used

### 5.1 Comparison of the Models

#### 5.1.1 AlexNet

Our model based on AlexNet performed the best among all the models tested achieving a training accuracy of 87% and a validation accuracy of 72% after 15 epochs. The learning rate was set at 0.1 and batch sizes for the training and validation sets were set at 32. The data set was run on a train-validation split of 70:30. The training loss obtained was also very low at 0.6.



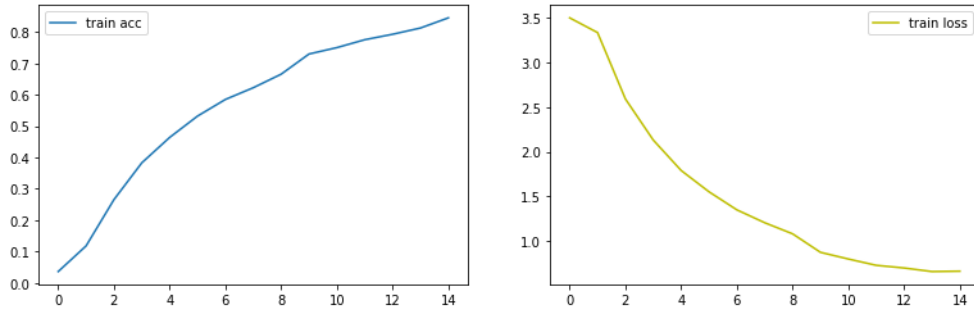


Figure 5.1: Training accuracy and loss of AlexNet Model after 15 epochs

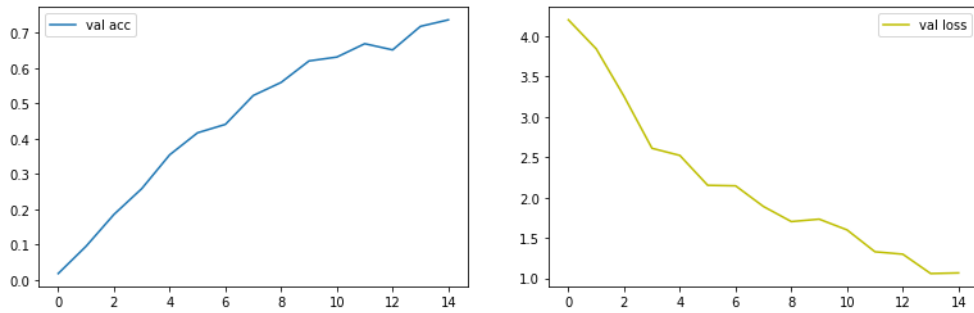


Figure 5.2: Validation accuracy and loss of AlexNet Model after 15 epochs

### 5.1.2 ResNet-50

ResNet-50 performed fairly well at a training accuracy of 78% and a validation accuracy of 62% after 15 epochs.

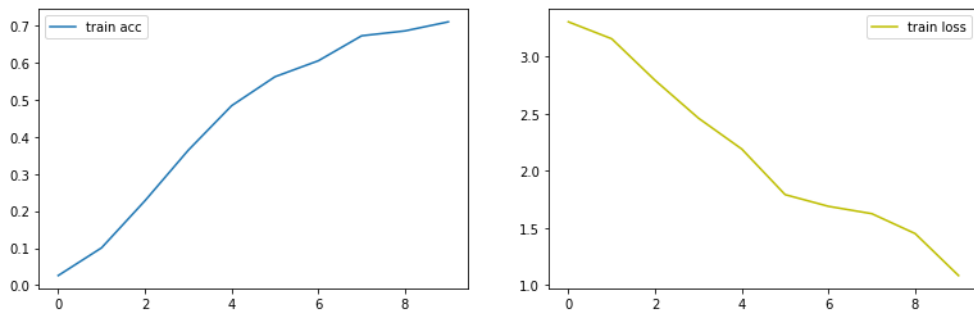


Figure 5.3: Training accuracy and loss of ResNet Model after 10 epochs

The same learning rate of 0.1 was set for ResNet with a train-validation split of 70:30. Train and validation losses were also low. Further epochs did not produce considerably larger accuracy rates.

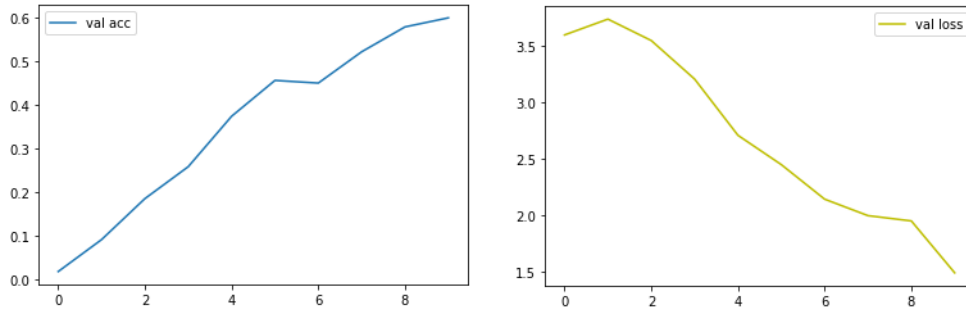


Figure 5.4: Validation accuracy and loss of ResNet Model after 10 epochs

### 5.1.3 SqueezeNet

The SqueezeNet model we trained, initially predicted to perform similarly to AlexNet, showed underwhelming performance. After 15 epochs it resulted in a training accuracy of 63% and validation accuracy of 49%. Tuning the hyper parameters did not result in any significant changes in the performance of the model.

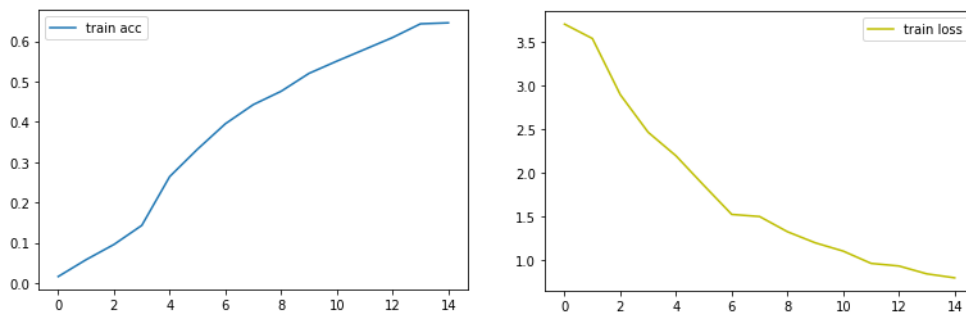


Figure 5.5: Training accuracy and loss of SqueezeNet Model after 15 epochs

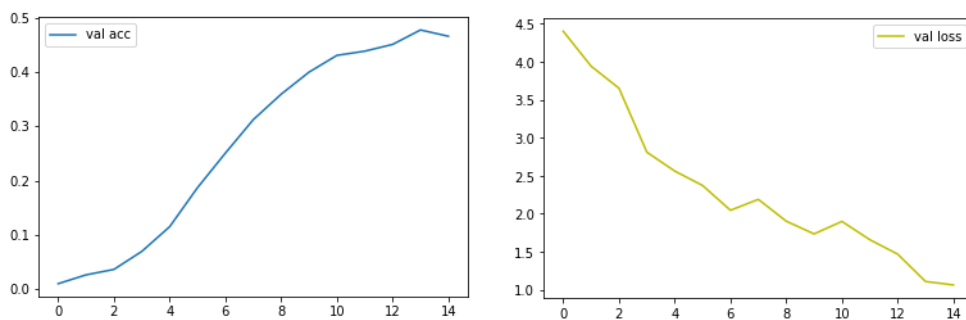


Figure 5.6: Validation accuracy and loss of SqueezeNet Model after 15 epochs

### 5.1.4 VGG-19

Our model based on VGG-19 performed poorly compared to the other models. The model rose to a training accuracy of 54% with a validation accuracy of 37% with little reduction in loss values.

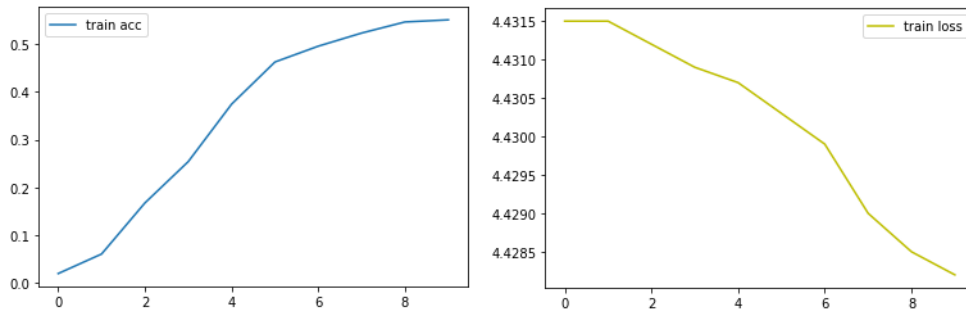


Figure 5.7: Training accuracy and loss of VGG-19 Model after 10 epochs

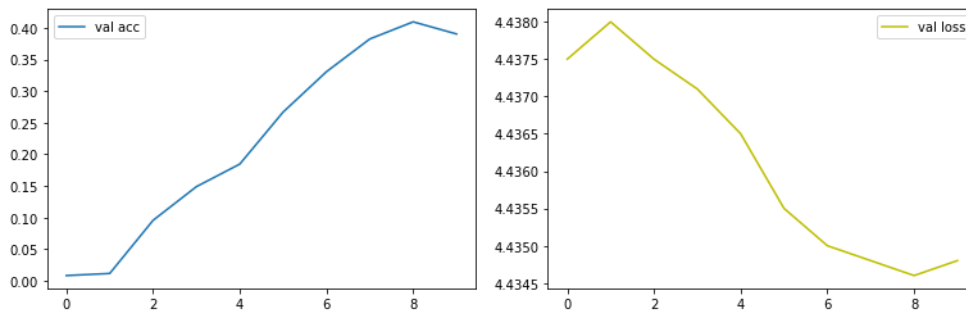


Figure 5.8: Validation accuracy and loss of VGG-19 Model after 10 epochs

## 5.2 Accuracy and Loss Values of the Models

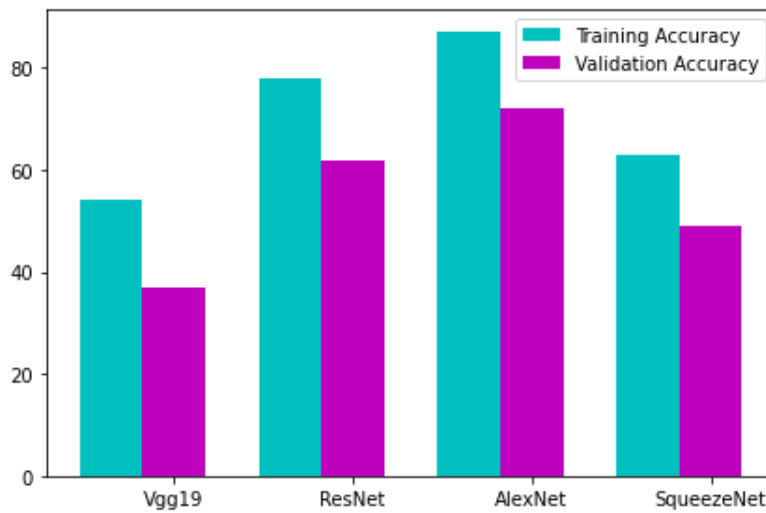


Figure 5.9: Relative Training Accuracy Achieved by the Models

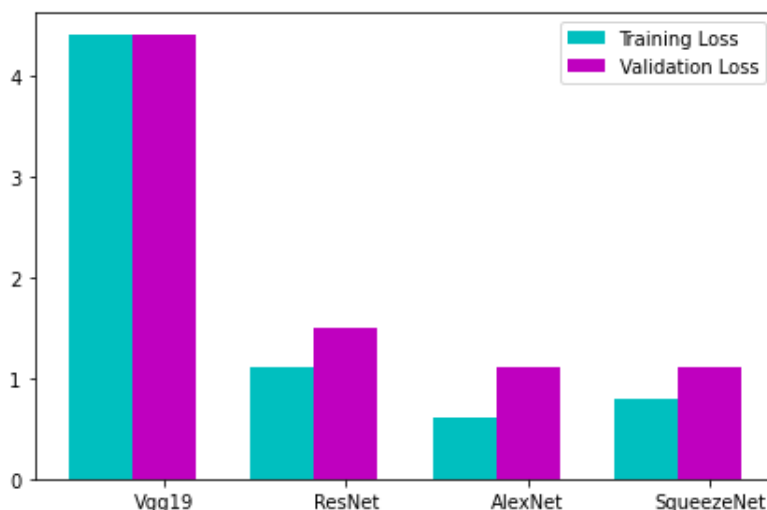


Figure 5.10: Relative Training Loss Achieved by the Models

### 5.3 Precision, Recall and F1 Accuracy

In this section we discuss a few other measures of accuracy we acquired from the two models with the highest training and validation accuracies. Since the data sets used for training and validation were supervised, or labelled, the model can compare its success in classifying the validation data accurately. The first measure is precision, which gives a measure of the ratio of correctly predicted positive data to the total predicted positive data: how many characters classified were actually classified correctly. Precision is measured as

$$Precision = TruePositives / (TruePositives + FalsePositives) \quad (5.1)$$

The second, recall, is a measure of the ratio of the number of data correctly classified for one class against the total number of data in the actual class.

$$Recall = TruePositives / (TruePositives + FalseNegatives) \quad (5.2)$$

The F1 score is a function of the precision and recall values and is generally a good measure of the accuracy of a data model. The F1 score is calculated as

$$F1 - Score = 2 * (Recall * Precision) / (Recall + Precision) \quad (5.3)$$

The data obtained for the ResNet-50 and AlexNet models are shown in Table 5.2.

Model	Recall	Precision	F1 Score
ResNet-50	0.51	0.75	0.61
AlexNet	0.59	0.81	0.68

Table 5.2: Recall, Precision and F1 Accuracy of ResNet-50 and AlexNet

## 5.4 Tuning Hyper-parameters

### 5.4.1 Train:Validation Split

Our AlexNet model was initially trained on a train-validation split of 80:20. Accuracy values obtained from the first computations were undesirable, as the validation accuracy was higher than the training accuracy, as shown in figure 5.11(a). We assumed this suggested that the data set put aside for validation was likely ‘less complex’ as opposed to the training set, giving the model an easier time validating. So the data set was re-distributed, this time with a train validation split of 70:30. After doing so, the model produced higher training accuracy as was originally expected (figure 5.11(b)).

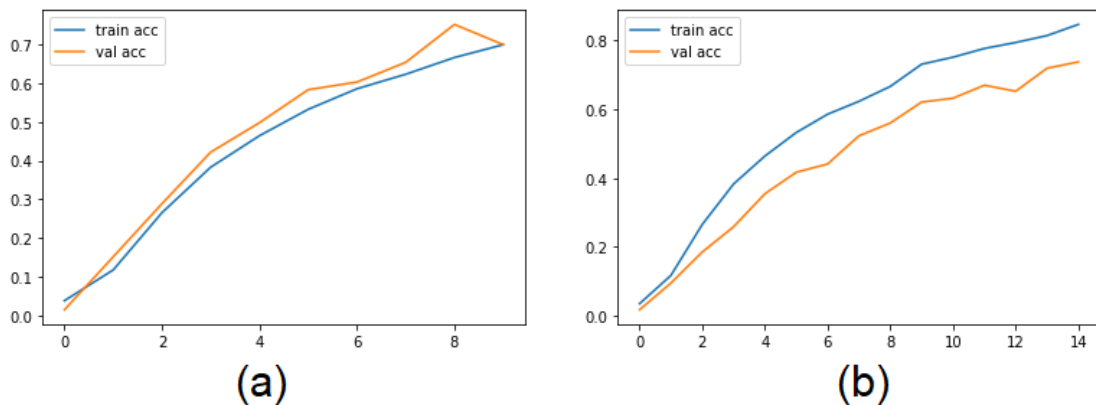


Figure 5.11: Accuracies of AlexNet on 80:20 split (a) and 70:30 split (b)

### 5.4.2 Optimizer

The model was implemented with Adaptive Moment Estimation (Adam) optimizer producing a training accuracy of 87%. Testing with Stochastic gradient descent (SGD) optimizer yielded a lower training accuracy of 84% at a learning rate of 0.1 and 15 epochs. The SGD optimizer, however, produced a slightly higher validation accuracy and was also significantly faster.

AlexNet Optimizer	Training Accuracy	Validation Accuracy	Average Time per Epoch
Adam (Adaptive Moment Estimation)	87%	72%	240s
SGD (Stochastic gradient descent)	84%	74%	150s

Table 5.3: Comparison of SGD and Adam optimizer on AlexNet

### 5.4.3 Input Size

The final accuracy and validation readings of each model were obtained with image input size of 100x100. The models were also tested with input sizes of 64x64 and 80x80. In general, the higher the input resolution, the greater the information the model has to learn and hence accuracy goes up. But this also increases processing time and requires greater computational capacity. While we were unable to use larger image sizes without a significantly powerful system, it can be assumed the performance can be significantly improved for all models using very high resolution data.

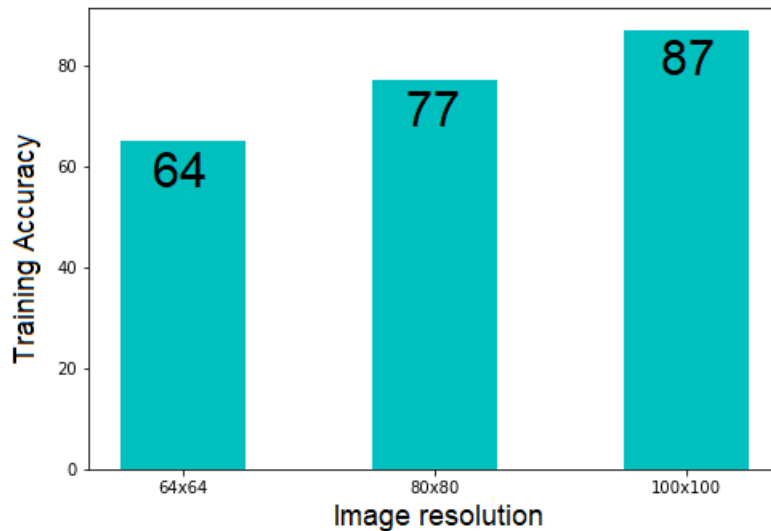


Figure 5.12: Variation in accuracy due to different input image sizes for AlexNet

# Chapter 6

## Conclusion and Future Work

Our goal initially was to convert Bengali handwritten characters to digitized editable text. But for now, we only predicted the alphabets using neural networks and compared the accuracy rates of different models. Research regarding Bengali language is developing and improving day by day. We are hoping that our work will open up a new field in the Bengali language based research areas and inspire many to do research on the Bengali language. We are planning to work on digitizing Bengali simple characters, compound characters, words, sentences, and numerals as well. A mobile app also can be developed by which anyone can gain easy access. Due to the pandemic situation we could not collect as much as data we wanted to. In the future, we are also hoping to increase the size of our data set and work on implementing other models and improve the accuracy rates. In the future we plan to convert handwritten characters to digital text and show the output in a notepad. In this research, we digitized Bengali handwritten characters so that it can be edited easily without a hassle when needed. Since most of the official documents in Bangladesh are handwritten, we are hoping that our work will be beneficial in this regard. Technological advancements have come a long way but research regarding Bengali language is still far behind. So, we are hoping that our work will motivate others to do research on this particular field and improve the existing works.

# Bibliography

- [1] Md. Zahangir Alom and Asari. “Handwritten Bangla Character Recognition Using The State-of-Art Deep Convolutional Neural Networks”. In: *Computational Intelligence and Neuroscience* 2018 (Dec. 2017). DOI: 10.1155/2018/6747098.
- [2] Tapan Kumar Bhowmik, Ujjwal Bhattacharya, and Swapan K Parui. “Recognition of Bangla handwritten characters using an MLP classifier based on stroke features”. In: *International Conference on Neural Information Processing*. Springer. 2004, pp. 814–819.
- [3] Mithun Biswas et al. “Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters”. In: *Data in brief* 12 (2017), pp. 103–107.
- [4] Jason Brownlee. *Understand the Impact of Learning Rate on Neural Network Performance*. Sept. 2020. URL: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.
- [5] R. R. Chowdhury et al. “Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation”. In: *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision Pattern Recognition (icIVPR)*. 2019, pp. 318–323. DOI: 10.1109/ICIEV.2019.8858545.
- [6] Sadia Chowdhury et al. “Bengali Handwriting Recognition and Conversion to Editable Text”. In: *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAEECC)*. IEEE. 2018, pp. 1–6.
- [7] Priya Dwivedi. *Understanding and Coding a ResNet in Keras*. <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33/>. Mar. 2019.
- [8] The Editors of Encyclopaedia Britannica. July 2017. URL: <https://www.britannica.com/topic/Bengali-language>.
- [9] Anshul Gupta, Manisha Srivastava, and Chitrlekha Mahanta. “Offline Handwritten Character Recognition Using Neural Network”. In: *2011 International Conference on Computer Applications and Industrial Electronics (ICCAIE 2011)*. IEEE, 2011.
- [10] SM Azizul Hakim and Asaduzzaman. “Handwritten Bangla Numeral and Basic Character Recognition Using Deep Convolutional Neural Network”. In: *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE. 2019, pp. 1–6.



- [11] Forrest N. Iandola et al. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and lt;0.5MB model size*. Nov. 2016. URL: <https://arxiv.org/abs/1602.07360>.
- [12] Puneet Puneet and Naresh Garg. “Binarization Techniques used for Grey Scale Images”. In: *International Journal of Computer Applications* 71 (June 2013), pp. 8–11. DOI: 10.5120/12320-8533.
- [13] Kaushik Roy and Shibaprasad Sen. “Towards Online Bangla Handwriting Recognition”. In: Jan. 2009.
- [14] Sourajit Saha et al. “Detecting Sex From Handwritten Examples”. In: *2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA)*. IEEE. 2018, pp. 1–7.
- [15] Isha Salian. *NVIDIA Blog: Supervised Vs. Unsupervised Learning*. Aug. 2019. URL: <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/#:~:text=In%20a%20supervised%20learning%20model,and%20patterns%20on%20its%20own>.
- [16] Synced. *ICLR 2019: 'Fast as Adam Good as SGD'-New Optimizer Has Both*. Mar. 2019. URL: <https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34>.
- [17] Rakshith Vasudev. *Understanding and Calculating the number of Parameters in Convolution Neural Networks (CNNs)*. May 2020. URL: <https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d>.
- [18] Jerry Wei. *AlexNet: The Architecture that Challenged CNNs*. Sept. 2020. URL: <https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951>.

# Appendix: Prepared Forms

অ	
উ	
ঐ	
খ	
চ	
ঞ	
ট	
দ	
ফ	
য	
ষ	
ড়	

আ	
ঊ	
ও	
গ	
ছ	
ট	
ণ	
ধ	
ব	
র	
স	
য়	

ই	
ঋ	
ঔ	
ঘ	
জ	
ঠ	
ভ	
ন	
ভ	
ল	
হ	
ব্	

ঈ	
এ	
ক	
ঙ	
ঝ	
ড	
থ	
প	
ম	
শ	
ড়	
বৎ	

Form 1 Page 1

বং	
ব্র	
বী	
বে	
ব্ধ	
ব্ধু	
ভ	
ভে	
ভ্র	
ভ্রু	
ভ্রু	

বঃ	
ব্	
বু	
বৈ	
ব্ধ	
ব্ধু	
ব্ধ	
ব্ধু	
ব্ধ	
ব্ধু	
ব্ধ	

বাং	
বা	
বু	
বো	
ভ	
ভ্র	
ভ্রু	
ভ্র	
ভ্রু	
ভ্র	
ভ্রু	

ব্য	
বি	
বু	
বৌ	
ব্ধ	
ব্ধু	
ভ	
ভ্র	
ভ্রু	
ভ্র	
ভ্রু	

অ	
ঊ	
ঋ	
খ	
চ	
ঞ	
ট	
দ	
ফ	
য	
ষ	
ড়	

আ	
ঈ	
ও	
গ	
ছ	
ট	
ণ	
ধ	
ব	
র	
স	
য়	

ই	
ঋ	
ঔ	
ঘ	
জ	
ঠ	
ত	
ন	
ভ	
ল	
হ	
ব্	

ঐ	
এ	
ক	
ঙ	
ঝ	
ড	
থ	
প	
ম	
শ	
ড়	
বৎ	

Form 2 Page 1

বং		বঃ		বাঁ		ব্য	
ব্র		বৃ		বা		বি	
বী		বু		বু		বৃ	
বে		বৈ		বো		বৌ	
ক		ক্স		ক্র		গ্ন	
ক্ষ		গ্ন		জ		জ্ঞ	
জ্ঞ		ট		ট		ঠ	
জ		ধ		ভ		ড	
জ্ঞ		ভ		ষ্ট		ণ্ড	
শ		শ		ক্ষ		শ্ব	
শ্চ		ক্ষ		ষ্ট		হ	
ক্ষ		হ					

Form 2 Page 2

অ	
ঐ	
ঐ	
থ	
চ	
ঞ	
ট	
দ	
ফ	
য	
ষ	
ড়	

আ	
ঊ	
ও	
গ	
ছ	
ট	
ণ	
ধ	
ব	
র	
স	
য়	

ই	
ঋ	
ঔ	
ঘ	
জ	
ঠ	
ত	
ন	
ভ	
ল	
হ	
ব্	

ঈ	
এ	
ক	
ঙ	
ঝ	
ড	
থ	
প	
ম	
শ	
ড়	
বৎ	

Form 3 Page 1

বং		বঃ		বঁ		ব্য	
ব্র		ব্		বা		বি	
বী		বু		বু		বু	
বে		বৈ		বো		বৌ	
ক		ফ		ফ্র		ফ	
স		ফ্র		ফ্রে		ফ্র	
ফ		ও		ও		ও	
ত্র		দ		দ্র		দ্র	
ন্দ		ক্র		ক্র		ক্র	
প		প্ন		ক		লট	
ষ্ঠ		ফ		ফ্র		ফ্র	
জ		দ					

Form 3 Page 2