An Analysis on Bengali Handwritten Conjunct Character Recognition and Prediction

by

Maazin Munawar
17101036
Yagghaseni Saha Roy
17101019
Mohammed Mudabbir Hussain
17101350

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
Spring 2021

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:

[illegible]

Maazin Munawar
17101036

Yagghaseni Saha Roy

Yagghaseni Saha Roy
17101019

Mudabbir

Mohammed Mudabbir Hussain
17101350

Approval

The thesis/project titled "An Analysis on Bengali Handwritten Conjunct Character Recognition and Prediction" submitted by

1. Maazin Munawar (17101036)
2. Yagghaseni Saha Roy (17101019)
3. Mohammed Mudabbir Hussain (17101350)

Of Spring, 2021 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)

Saiful

Md. Saiful Islam
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

[illegible]

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

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Mahbubul Alam Majumdar, PhD
Professor
Department of Computer Science and Engineering
Brac University

Abstract

In the very active field of handwriting recognition, a lot of research can be found in the detection of the handwriting of various languages, especially English. However, for languages like Bengali, while they hold some success in handwritten character recognition, a big roadblock is Bengali conjunct characters or “Juktakkhor”. As Bengali conjunct characters are very complex, even today many institutions in Bangladesh still maintain documents as handwritten copies. In this paper, we will present a model that focuses on conjunct character recognition and conversion to text format. Our proposed system will be trained and tested using CNN models like VGG19, ResNet-50, GoogleNet, LSTM, ShuffleNet etc. The results generated from preliminary analysis yield that ShuffleNet gives the most accurate results with an accuracy of 91.2% followed by GoogleNet with 73.3%.

Keywords: Conjunct Characters, Handwritten, Neural Networks, Bengali, Segmentation, ResNet-50, ShuffleNet, LSTM, GoogleNet.

Acknowledgement

One of the main factors which made this thesis possible was the continuous support and guidance of our supervisor, Md. Saiful Islam. He provided us with adequate materials to research and study Convolutional Neural Networks as well as various Image Processing Techniques. With his guidance we were able to conduct the research in a structured and organized manner. We would like to thank Mr. Babul Kumar Saha Roy all the people who assisted us in collecting and acquiring our data. We also thank all the authors whose research has played a huge role in this field and has helped us greatly in conducting our own research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Bengali, also goes by its endonym Bangla, is seventh in the hierarchy of most spoken languages around the globe [21]. Despite the fact that, Bengali language has predominance in Bangladesh and Indian subcontinents, there are not many contributions in the research works related to digitizing handwritten documents in Bengali to editable text format. Handwritten documents have existing importance in most of the vital official, legal and academic institutions. In this era of evolving cutting-edge technologies, those documents are preserved in the form of hard copies. The formal forms, literary works, registers, archives and all such records are prone to the disadvantages of storing them as paper copies, which take up a lot of storage space. Keeping backups or even making necessary edits results to be a laborious task in this regard. Therefore, to make procedures smooth, development of a handwriting recognizer would be very beneficial as a whole. The documents could be converted to digital and editable format effortlessly. Moreover, updating or correcting, and also machine sorting or automation of digital documents would become feasible. This paper presents an evaluation of algorithms and models to carry out the technique of recognition and conversion of Bengali handwritten characters to digital format. We have implemented several CNN models for testing our system.

1.2 Problem Statement

In current times of technological advancements, digitization is the key to system efficiency. Easy access and management are necessities to keep up with the tide of progress. Bangladesh, such a densely populated country, is required to gather and store huge amounts of official and personal information about people. Even today, many government official, educational and medical institutions maintain crucial data in the form of hard copies. Real estate ownership, administration legislative files, academic registry, medical records and many more, all are to be found in handwritten format and most of these are in Bengali. Digitizing these documents would need to be done manually, which will result in inefficiency, consuming both money and time. A recognition system such as our proposed one would bring about a catalytic effect in the procedure. Unprecedented management errors or inaccessibility in such documents could create a deficiency in the organizations.

Physical transcripts are prone to be used for fraud, adulteration and misplacement. These kinds of situations would raise a question on the credibility of the system. Strict maintenance or supervision would tend to be incompetent and a loophole could be easily created in the system.

Furthermore, The task of keeping and maintaining physical copies is a waste of space and labor. So a system to digitize those physical copies would help in many areas. This can help in many instances, such as in online banking. Cheques with signatures in Bengali can be processed digitally and remotely without going through the hassle of going in person. There are many hospitals that keep their records as hard copies, which makes the process of manually inputting data in the database very cumbersome. Having a system that can accurately convert the records to editable values by simply feeding images to the system is efficient. In other areas of cataloguing and ledgering, like in libraries, such a system would also be useful.

Therefore, an easy and efficient approach is necessary for recognition and digitization. This will greatly reduce workload of the management and also improve service quality received by the general public.

1.3 Aim of Study

The aim of our study is to implement a system that identifies and transforms Bengali handwritten documents having conjunct characters into digital editable text with high accuracy. Existing research works have been found in this field, but the objective is not fully satisfied. Most of them perform the conversion on simple handwritten Bengali characters or attain significant results with printed rather than handwritten texts. The core of our system will be working with handwritten conjunct characters as well as when they are conjoined with diacritics.

Google Cloud Vision API, is presently an available system for digitizing handwriting text documents [25]. This system can work on different languages to extract text from documents and images with high accuracy, but it has a slight drawback in the detection of handwritten Bengali conjunct characters. Another available technique is an online-based Optical Character Recognition (OCR), the i2OCR [22]. This can give editable texts from images and scanned documents, but the input is required to be in printed format. Our primary focus was to fill in the void.

1.4 Research Methodology

The ultimate goal is to recognize Bengali Handwritten text with simple and conjunct characters and transform it to digital text. The first challenge of the research for us was to select our data set forms, because the chosen combination of data plays an important role in this. We made a target to collect versatile data for our system i.e. conjunct characters with and without diacritics. Handwriting has variations of different styles and strokes, and it greatly differs from person to person. That is why we intend to collect a data set with a massive divergence in handwriting techniques and unpredictable user-defined alphabets which are required to train our system.

We have had a focus group discussion and identified where data is needed. We came up with three different forms each with different sets of conjunct characters for our data collection. Our aim was to collect about 100000 data on conjunct characters. But unfortunately, we were able to collect about 25000 raw character data. In these extraordinary situations of the Covid-19 pandemic, it was not possible for us to carry out data collection procedures as we have planned, yet we were able to manage a moderate sample for our research.

We collected our data remotely from known people who had an access to scanners so that we could conduct our research while maintaining safety protocols. The data is far away from what we required, but were restrained to start working with whatever was managed. The raw data was converted to image form by either using a scanner or mobile camera. The characters are extracted from the data set forms and the data is further processed for smoothening by resizing and gray-scaling to make it more recognizable for pre-processing. Techniques of augmentation and then binarization is also applied. A considerable difficulty in recognition of conjunct characters is due to size and variations present in people writing the same character in different ways. Some conjunct characters have lines and curvatures which have similarities with other conjunct characters and even with some other basic characters, this adds another level of complexity.

For example, the conjunct character ব্দ is similar to the basic character দ (shown in Figure 3.1). Another obstacle is to categorize those characters that don’t look like the basic characters they are made up of, like the character ক্ষ, which is made up of ষ and ক.Also different people have different styles for writing the same thing. Some people write ক্ত also as ক on ত.

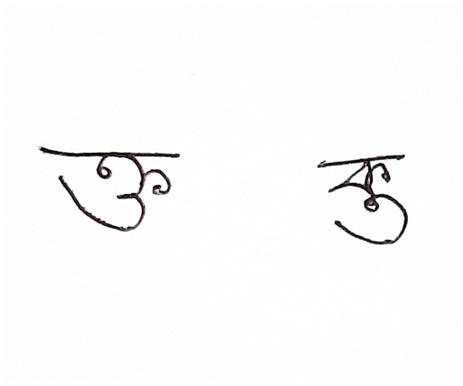


Figure 1.1: Shows how conjunct characters increase in complexity with different styles of representation.

Then the data is tested by running through several fine tuned Convolutional Neural Network (CNN) models, such as VGG19. ResNet-50, LSTM, GoogleNet and ShuffleNet. An analysis of accuracies obtained is then carried out as our concluding part.

1.5 Thesis Outline

The overall report outlines our motivations in conducting the research, the procedures and protocols we followed and finally our results and findings.

This report is organized as follows with Chapter 1 being the introduction which includes the motivation, objectives and methodologies of our paper along with the problem statement.

Chapter 2 is the literature review section where papers that have dealt with similar topics have been discussed. All the discussed papers and journals helped us immensely in learning about all the methods and models that we have used in our research.

In Chapter 3 we have discussed the outline we had planned for the research as well as how we had envisioned to process the raw data. We discuss in detail about various pre-processing methods and why we chose each one.

In this section (Chapter 4) we dictate the selected models for our research and their implementations. We discuss what sets the models apart from each other as well as models that we have not used in conducting the research.

Finally, in Chapters 5 and 6, we discuss our findings and results. Which model performs most significantly for our data is deduced along with future scopes for study.

Chapter 2

Related Work

The analysis and detection of handwritten characters has become more accurate and prominent over the past few years and many researches are still currently taking place. In Hakim and Assaduzzaman's work [12], the data set used was BanglaLekha-Isolated data set and a further 6000 data sets were collected for cross-validation. The samples are converted to binary by the Otsu's method and other methods such as Gaussian filter and augmentation techniques to further pre-process the data. Then a modified version of VGG-16 CNN was used on the samples which yielded a 95.25% and 99.6% accuracy on basic characters and numerals respectively. However, their system is limited to basic characters and numerals and exclude conjunct characters.

In another approach, Chowdhury et al. [9] used supervised learning for character recognition. They collected their own data set, 2000 handwritten samples per basic Bengali Character. 700 samples per character was used for training to avoid noisy images and the carefully processed so size and resolution was the same for all images. The neural network system used was trained using Scaled Conjugate Gradient Backpropagation and about 85 features were extracted per character which was stored in a separate matrix with 85 columns, as such each row of features corresponds to a single alphabet. The system was trained using MATLAB and the data was trained several times to minimize error and post-processed to remove noise. The accuracy of their system ranged from 85% to 95% per letter. A limitation of their model was that it did not include conjunct characters or Bengali numerals.

On the other hand, in Gupta, Srivastava and Mahanta's [2] created an OCR for the English language by comparing different classifiers. The data was slant corrected and segmented using heuristic algorithms. The pre-processed data was put through a multilayer feedforward neural network with backpropagation algorithm and Fourer descriptors were used for feature extraction. The data was trained and tested using MLP, RBF and SVM and ultimately the final system was constructed using SVM as a classifier as it gave comparatively better results. Their OCR was able to correctly identify 21/26 word images. The result and complexity would be different if the system was trained and tested for Bengali conjunct characters.

Saha et al.[10] made a sex determination system based on handwriting. They used the BanglaLekha-Isolated data set. They resized and rotated each sample by varying degrees to increase sample size by five times and reduce risk of data dependency.

After which, loss function was regularized with Frobenius Normalization and Adam optimization algorithm was used for fast optimization. A DCNN model to classify alphabets by using Tensorflow and Tesla K80 GPU to build and train their five models (VGGNet, GoogleNet, ResNet, FractalNet, and DammNet "in press"). The DammNet "in press" performed the best overall with the most accuracy (97.21%)and least amount of parameters. The CNN model is then auto encoded and fed into RNN model trained on Tesla K80 GPU for sex prediction with 91.85% accuracy.

Bhowmik, Bhattacharya, and Parui [1] created their own Bengali character database with 25000 samples. The training, validation and test sets consist respectively of 350, 60, and 90 sample images for each of 50 Bengali basic characters, conjunct characters and Bengali numerals were not included. They used a MLP classifier based on stroke features for character recognition. Their proposed model identifies strokes as digital curves with 1-pixel width and 10 features (such as size, shape, and position information) are extracted from each curve and then normalized. The feature vectors of the strokes are concatenated in a particular order to form the feature vector with a length of 100 of the character image. The feature vector is then fed to appropriate MLP classifiers for final classification. They ran two different simulations, one using 100 input features computed from east and south projections and another having 200 which was computed from all four directions. For each case, several simulations were run changing the number of hidden layer nodes. It was observed that recognition accuracy was highest for both training and test set when the hidden layer size was half of the feature vector sight.

Rahman et al.[5] proposed a CNN based model for Bengali handwriting recognition. The proposed method first normalizes the written character images and then employs CNN to classify individual characters. They prepared a data set with 400 samples per basic character, conjunct characters and numerals not included. The data is resized and converted to gray-scale. No traditional methods for feature extraction was performed. The data was directly fed to their proposed BHCR-CNN since convolutional operations have the ability to extract features and this results in a certain degree of shift and distortion invariance and fewer numbers of free parameters used. 17500 and 2500 samples had been used for training and testing respectively. Their system shows correlation between learning rate(LR), batch size(BS) and recognition accuracy. The most accuracy achieved was 93.93% and 85.36% for training and test sets respectively at a LR of 1.0 and a batch size of 10.

Siddique et al.[18] have proposed a system for English to Bengali machine translation where they used a recurrent neural network. They collected about 4000 data samples from articles for both English and Bengali and maximum length per sentence was 7 and 8 respectively. About 80% of the samples were used for training and the rest for testing. The input data is tokenized and then vectorized by multiplying encoder outputs and attention weights. Context vector is calculated using sigmoid and softmax functions. Both GRU and LSTM are used for performance evaluation. GRU performed better than LSTM. Adam optimization algorithm was used for learning data sets. They used different activation functions and the best result yielded from linear activation function in the encoder layer and the tanh activation function in the decoder layer.

Bag and Harit [3] made a comprehensive survey about the different works on OCR for Bengali and Devanagari scripts. The papers selected for their survey work with a range of input from printed character and numerals to handwritten character and numerals, along with conjunct characters. They made a comparison table of all the reported methods. They compared each surveyed work with respect to feature set, classifier, and reported accuracy. They have also reviewed post-processing methods that are used to improve accuracy of OCR technology. They made four sets of tables, each set having a table for both Bengali and Devanagari. The sets were organized according to input. One set for printed character and numeral, one for handwritten character, one for handwritten numerals, and the last for conjunct characters. For compound characters, the highest accuracy was found to 86.10% which was achieved by implementing a topological feature set and template matching classifier with 50 output classes and 1000 training sets.

Rumman et al.[13] proposed a system in their paper that recognizes and converts isolated Bengali characters to digitally editable format. The goal of their model was to train a large amount of data and classify each sample in real time. They used a CNN model that was implemented using Keras with Tensorflow as backend. The CNN model contained two convolutional layers each with 32 filters and a 5x5 kernel size, and the Relu activation function was used. Each convolutional layer is followed by a maxpool layer with 2x2 pool size. Then there are three fully connected layers with 1024 nodes and Relu activation function. Lastly, is the output layer with 50 nodes and Softmax activation function. The model is optimized using SGD. The system was implemented on Google Colab. The data set used was BanglaLekha-Isolated data set. Around 2000 samples of handwriting was collected for each character. The data was augmented using the ImageDataGenerator function from Keras library. The training process was done on the base data set and then repeated on the augmented data set. The model was applied to similar data sets to check versatility. The model achieved 91.81% accuracy on BanglaLekha-Isolated. The accuracy achieved on the test set was 95.25%.

Alom et al.[8] evaluated different state-of-the-art DCNN models on HBCR application on Bengali numerals, alphabets and special characters. The models used were VGG Net, All-Conv, NiN, ResNet, FractalNet and DenseNet. The experiment was performed with Intel Core-I7 and Keras with Theano on the backend. The DCNN models were evaluated on three data sets from CMATERdb (6000 samples for numerals, 15000 for alphabets and 2231 for special characters). For handwritten digit recognition, testing accuracy was best with DenseNet at 99.13% with FractalNet a close second. DenseNet also showed the most accuracy for handwritten alphabets with 98.31% accuracy followed closely by FractalNet and VGGNet. For conjunct characters DenseNet again had the best accuracy at 98.18% with FractalNet at 97.98%. DenseNet had around 4.25M parameters, less than VGG net (8.43M) and FractalNet (7.84M). DenseNet also took the most computational time per epoch. Lastly the DCNN models were compared against existing methods and the results show that DCNN models fare better compared with other methods.

In Roy and Sen's paper [23], a procedure to recognize online Bengali handwriting recognition is explored. For their model, 59 basic strokes were used for recognition purposes for feature extraction. A two-phased neural network is used. For online handwriting recognition, pen tip movements and positions are the data (known as digital ink). They used Wacom Tablet, A4 take note and the data sheets for data collection. pre-processing techniques such as substitution, removal, reordering, and extraction of the data was applied. Segmentation was done keeping in mind the concept of downward stroke for Bengali handwriting. The downside movement is where the stroke is split which is in the upper part of the image. A total of 233 features are used for recognition, 15 of which are structural, 90 are point based and the rest 128 are directional features. When the stroke goes through the neural network, basic strokes of the input word are identified in first phase and in the second phase the actual character id of the corresponding strokes are determined. Their model recognized the word ফল 19 out of 50 instances.

Work Reference	Public Dataset	Collected Dataset	Model	Accuracy Public Dataset	Accuracy Collected
Hakim et al. [10]	BanglaLekha-Isolated	6000	CNN	99.44%	95.25%
Chowdhury et al. [8]	n/a	35000	ANN	n/a	95%
Bhowmik et al. [1]	n/a	25000	MLP	n/a	84.33%
Rahman et al. [5]	n/a	20000	CNN	n/a	85.36%
Saha et al. [15]	BanglaLekha-Isolated	200	DamnNet "in press"	97.21%	n/a
Rumman et al. [11]	BanglaLekha-Isolated	n/a	CNN	91.81%	n/a

Table 2.1: A comparison among previous works.

Chapter 3

Data Analysis

3.1 Workflow

To make sure we can conduct our research in a systematic manner, we laid out a structured plan which would help us achieve our goal. At the early stages of our research, we created forms which we then distributed to our family, friends and peers to collect our testing data set.The same Bengali conjunct character is written differently by many people, hence it was essential that our data set covers most of these variations. As there are approximately 90 conjunct characters, we divided our forms into three parts, making sure that together, most of the commonly used conjunct characters were included. Our training data set was decided to be the BanglaLekha Isolated data set. [6]

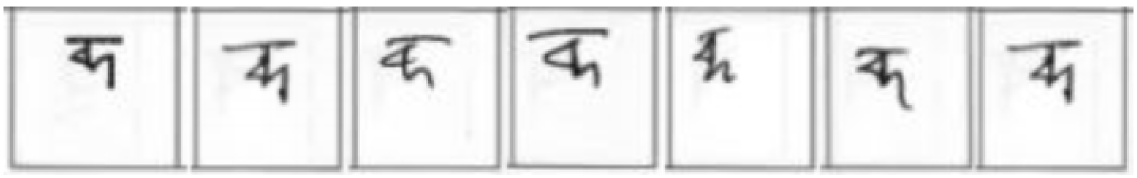


Figure 3.1: Shows how the same conjunct character is written differently by people.

After collecting all the data, we ran them through a series of binarization and thresholding techniques. They were then fed into several Convolution Neural Networks (CNNs) and fine tuned to achieve our desired results.

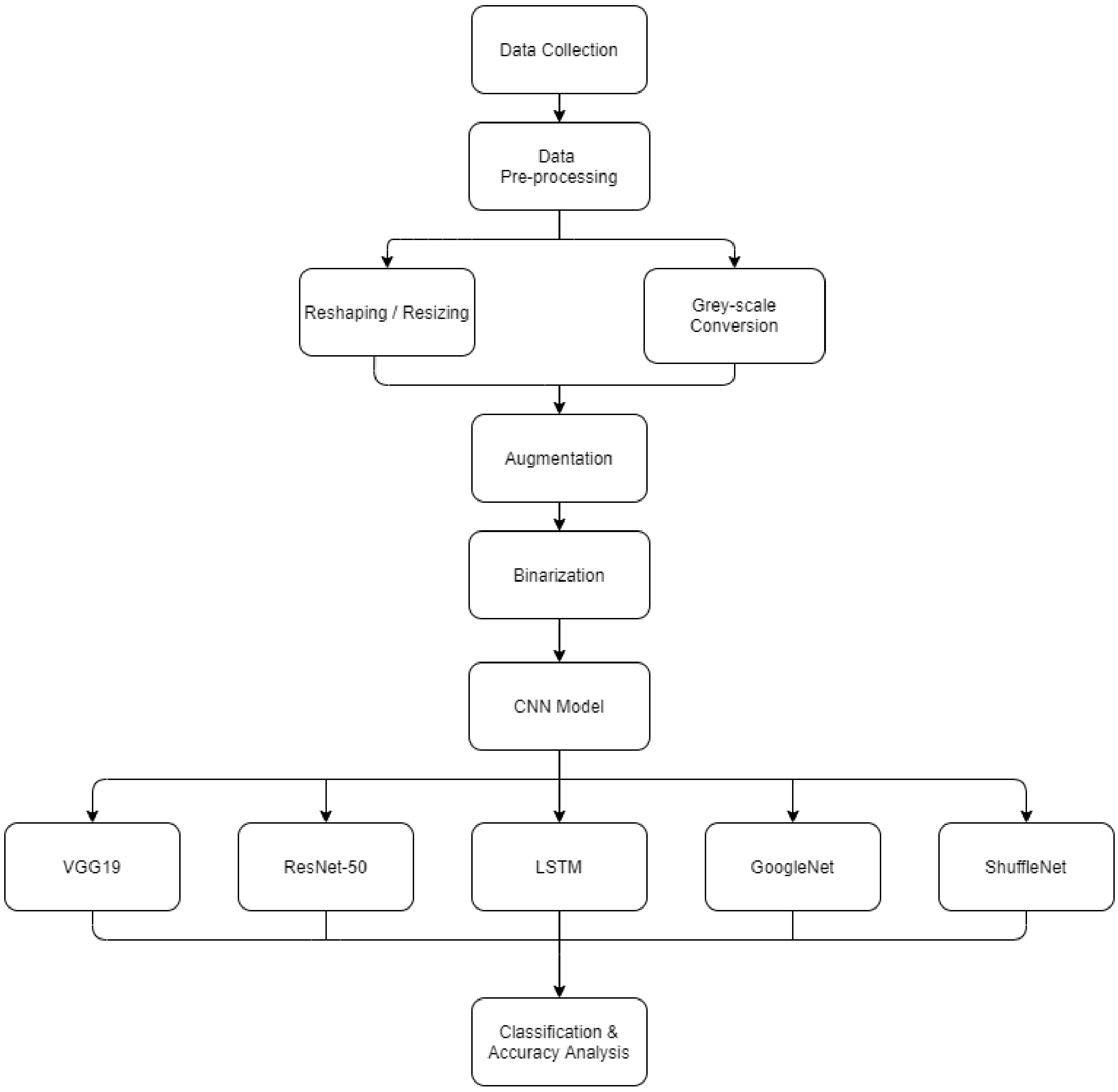


Figure 3.2: Shows the workflow followed in the research.

3.2 Data Collection

One of the many hurdles of making a system such as this is that there are not enough data sets easily available. Some of the well known databases for Bengali Handwriting such as Ekush [19] and ISI [20] is shown in Table 3.2

The secondary data set to be used for training, cross-validation and testing is BanglaLekha-Isolated [7] which is a publicly available multi-purpose comprehensive data set of Handwritten Bengali Isolated characters. It contains 84 characters (50 basic characters, 10 Bengali numerals and 24 selected conjunct characters) and about 2000 pre-processed samples of each character.

Datasets	Characters
BanglaLekha Isolated [7]	166,105
Ekush [19]	673,482
ISI [20]	30,966

Table 3.1: Existing data sets.

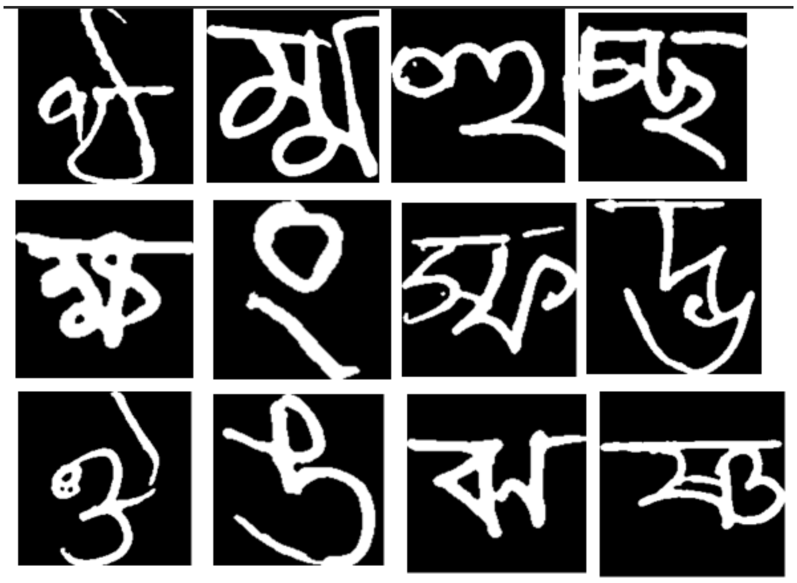


Figure 3.3: Some images from BanglaLekha Isolated.

Furthermore, for real world validation, we have collected 300 handwriting samples so far to further justify our proposed model. Existing data sets such as BanglaLekha Isolated [7] have limited variations in conjunct characters and they have no data on diacritic forms. Hence, we collected data to expand the number of variations in conjunct characters as well as gather data on diacritics. We have created three different forms for data collection and the 90 conjunct characters have been divided into the three form i.e. 30 conjunct characters per form. Data on basic characters are also present in each form. All the forms have the same 46 basic isolated characters (11 vowels and 35 consonants). The last four consonants are adjoined to the character “ba” (ব). An additional 10 vowel diacritic forms appended to “ba” (ব) are also included.

3.3 Pre Processing

Image processing requires a lot of GPU/CPU power to run smoothly. If RGB images are fed into neural network models, it takes significantly longer for the model to be trained as the images each have 3 channels. The resolution of the images also needs to be taken into consideration as it increases the number of pixels the model needs to go through for each image. Hence, it was essential that the test and train

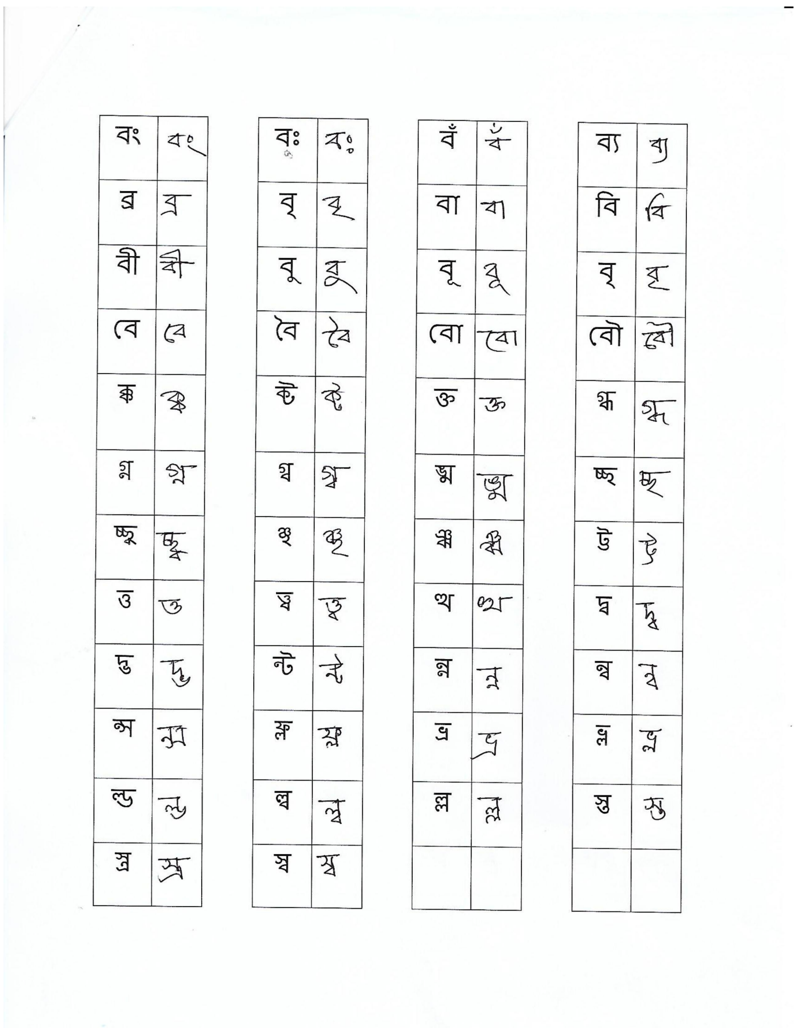


Figure 3.4: A snippet of the form used to collect raw data.

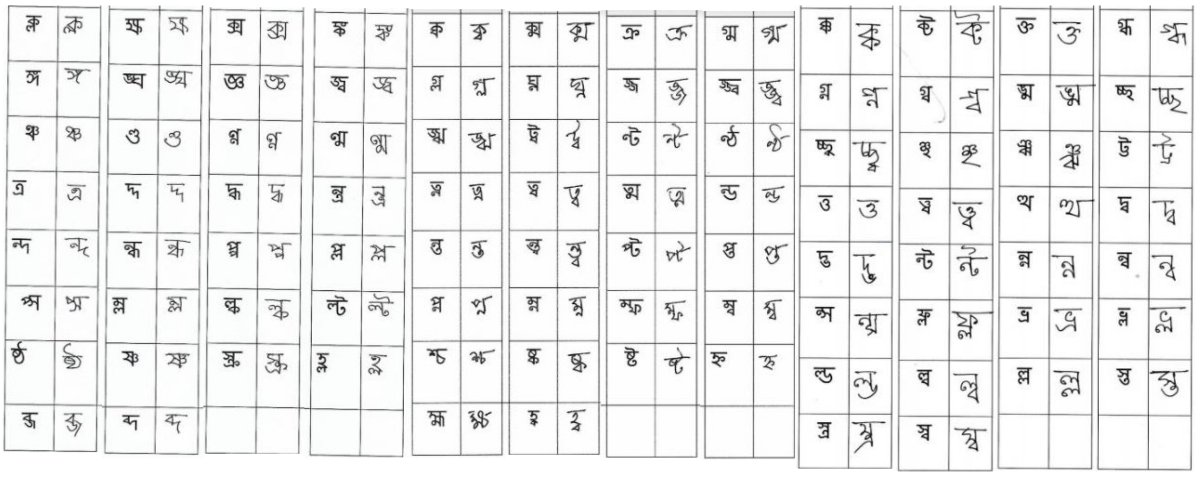
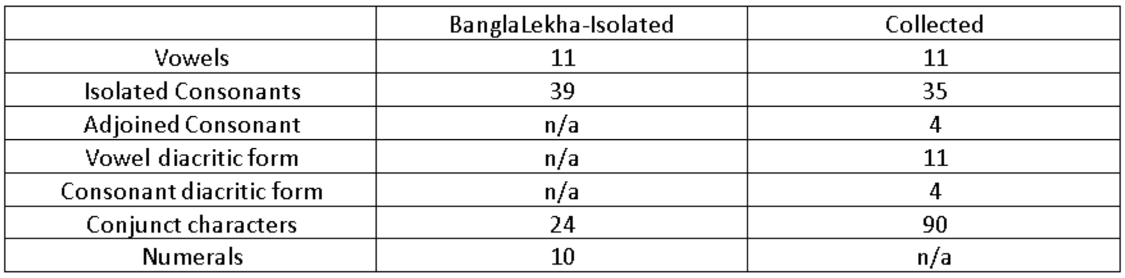


Figure 3.5: A compilation of conjunct characters from the raw data.



	BanglaLekha-Isolated	Collected
Vowels	11	11
Isolated Consonants	39	35
Adjoined Consonant	n/a	4
Vowel diacritic form	n/a	11
Consonant diacritic form	n/a	4
Conjunct characters	24	90
Numerals	10	n/a

Table 3.2: Summary of sample data collected.

data sets are pre-processed to make it easier for the models to learn.

3.3.1 Resizing and Grayscaling

The data forms were scanned at 200dpi with dimensions 1700x2200. This is still too large for the models to process easily. Using python's OpenCV library, we resized the images to half their original resolution before moving on to binarization.

Binarization requires the images to have only one channel, meaning the images need to be in gray-scale format before they can be binarized. We converted each image to gray-scale which significantly reduced the computational power required by the models to process each image as there is only a single channel that it needs to process.

3.3.2 Binarization Techniques

In OCR, some of the most important algorithms used are part of binarization.. They convert gray-scale images that contain 0 to 255 levels of gray into black and white images (0s and 1s). However, it is difficult to choose the best binarization technique as they vary greatly in structure and complexity. Binarization methods can be broken down into two categories, Global and Local binarization. Global binarization uses a single threshold value and applies it to the whole image whereas local binarization calculates a threshold value for each pixel using the surrounding pixels. We used a few of these binarization methods to deduce which one best suits our needs.

3.3.3 Otsu's Thresholding

Otsu's thresholding computes and applies a single threshold value on the whole image. This is not effective on images with varying levels of illumination. As we can see from figure: 3.7, the output image has no distorted borderlines and appears to be quite clear. As the image consists of not many colors or varying illumination, a single global threshold applied to the whole image performed well.

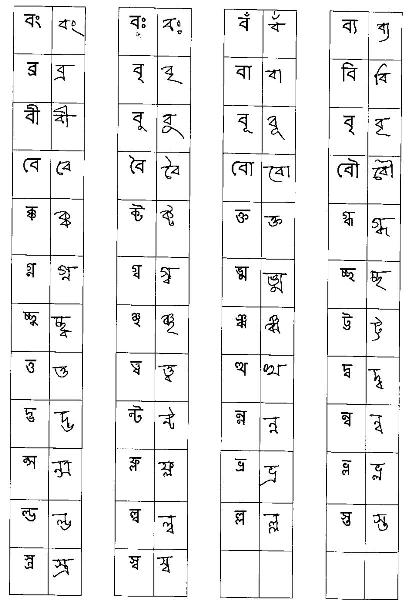


Figure 3.6: Binarized

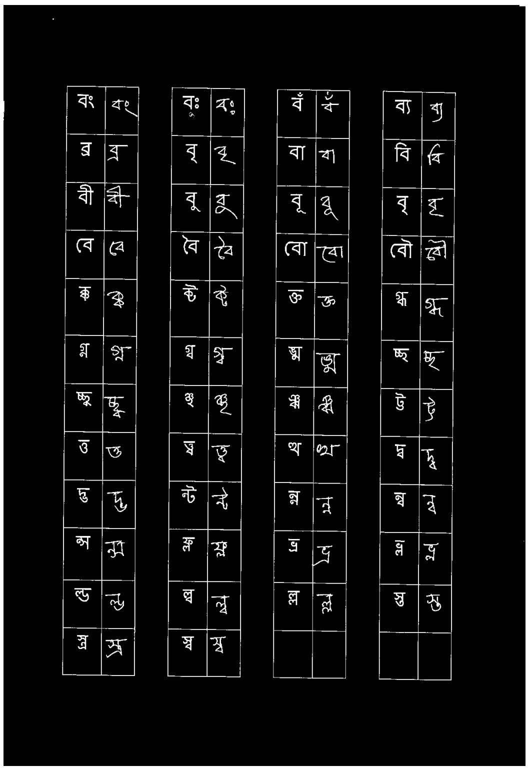


Figure 3.7: Binarized and Inverted

3.3.4 Adaptive Thresholding

Adaptive Thresholding on the other hand is a local binarization technique. A filter or window of size NxN is slid over the whole image and the threshold for each window is calculated and applied to the pixels that are within each instance. This gives a much more accurate result as regions with varying intensities of light are also accounted for.

Figure: 3.9 shows that the output generated after using Adaptive Thresholding is much lower in quality than the previous one. There are distorted and broken lines in the borders and the whole image appears to have black pixels all over the background. As it calculates a threshold for each region, any lighter shade of black which may be have appeared on the image might be turned to white completely or vice versa as opposed to other regions where the black is more profound. This disparity can be clearly seen in figure 3.10.

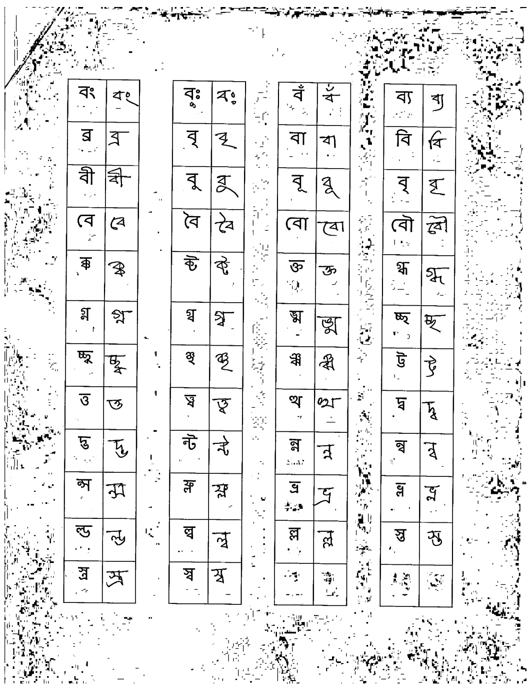


Figure 3.8: Binarized

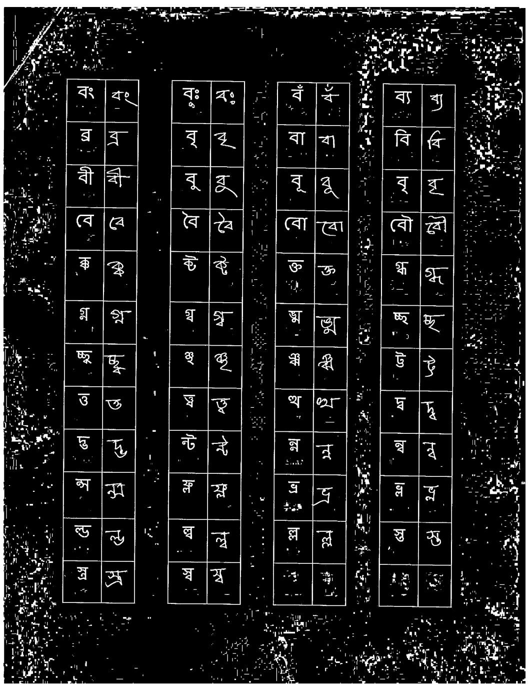


Figure 3.9: Binarized and Inverted

Methods such as the Kilter and Illingworth Method [4] which divide the image into foreground and background was also considered. It was later discarded as our research only consisted of handwritten characters and did not require such segmentation. For our research we chose Otsu's Thresholding out of all the techniques as it gave us the most promising results.

3.4 Bounding Boxes and Character Extraction

To extract the characters from the collected forms we used a bounding box method. A high resolution image of the form was passed into the algorithm which would then draw boxes around the characters. Initially, the algorithm was drawing boxes which held two characters, but these were unusable as the CNN models needed to be trained using single character images.

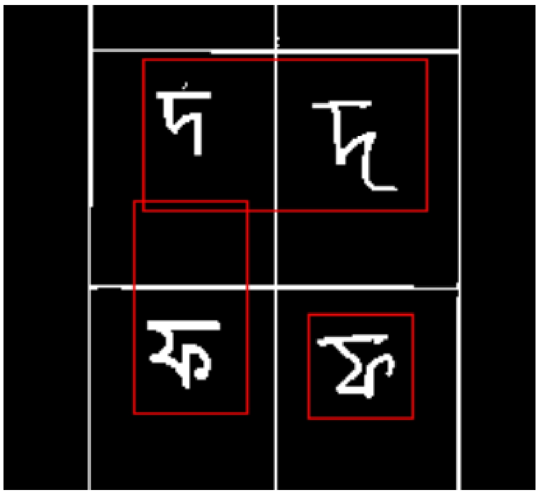


Figure 3.10: The bounding box algorithm taking in two characters in a single box.

To solve this problem we added a simple clause to the algorithm which restricted it to only draw boxes that were under a predetermined threshold area value. A simple loop is then executed, using the pixel values of the edges(bounds) of each box to crop out the characters from the forms, ready to be used with the CNN models.

Figure 3.11: A single character image extracted from the data forms.

3.5 Augmentation

As it was difficult for us to collect a large number of data due to the global pandemic, an efficient way for us to artificially increase the data size was augmentation. Having a large data set is essential as it helps neural network models learn better. Augmentation can be of various types such as rotations, horizontal or vertical translations, shear, changing brightness levels, etc. Rotations would completely change the handwritten character hence we omitted it. Vertical translations and shear on the other hand would give the models some new data to learn from while maintaining a standard for the characters. A combination of the python libraries Pillow and OpenCV were used to achieve the augmentations.

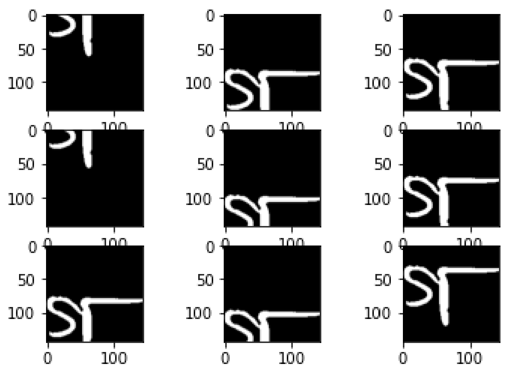


Figure 3.12: Vertical Augmentation

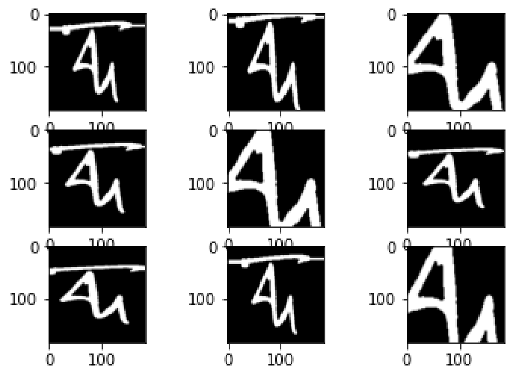


Figure 3.13: Shear Augmentation

Chapter 4

Model Implementation

4.1 Convolutional Neural Network

In the past years the field of Artificial Intelligence has made enormous strides in hopes to be able to lessen the gap between the abilities of humans and machines. One of the main algorithms that has made Photo and Video Analysis, Image Classification and Language Processing possible is a Convolutional Neural Network. This is a deep learning algorithm that is able to take images as input. It assigns weights and biases to various properties of the image and is able to differentiate between them. The architecture of a CNN is similar to that of the connections of Neurons inside the human brain. It has convolutional layers, pooling layers, padding layers, filter layers and concatenating layers, some of which have trainable parameters and have weights assigned to each parameter. It is able to detect features such as shape, illumination, area, etc. from images by applying the correct filters. Some of the renowned CNNs of today are AlexNet, GoogleNet, ResNet and ShuffleNet, a few of which have been used in this research. Figure 4.13 shows how a typical CNN structure processes handwritten characters [11].

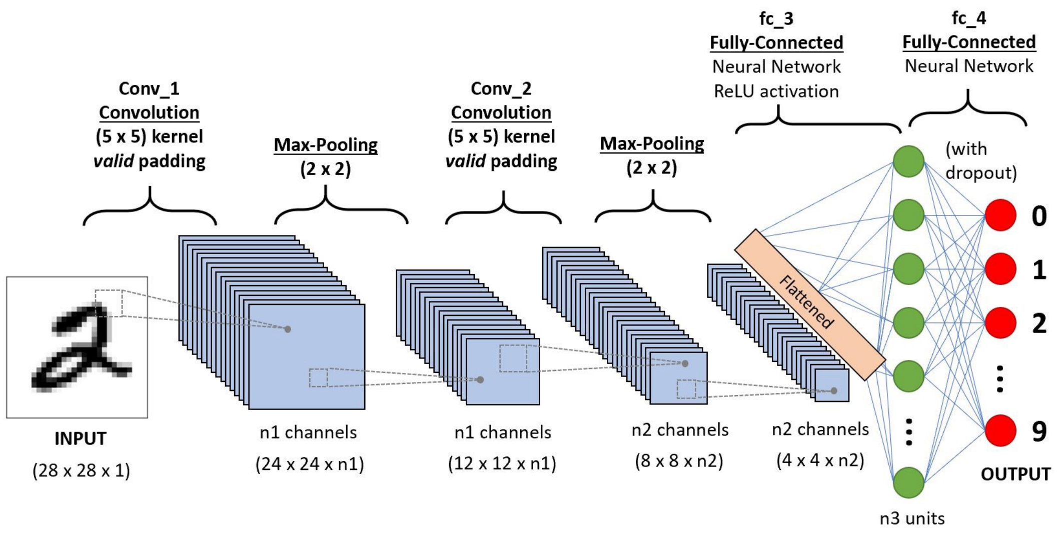


Figure 4.1: How a CNN classifies a handwritten character.

4.2 Supervised Learning

In machine learning, algorithms are required to be trained. A sub category in this is called Supervised Learning. The algorithm is trained but it requires the data set to be labeled. In our research the training data is labeled as we require the models to be able to deduce which class an image belongs to. Supervised learning does have some drawbacks. It is very easy to overfit the data. A data sample may be incorrect but also have the same labels as a correct data. This may also give inaccurate results.

4.3 Optimizer

One of the most influential components in a CNN is the optimizer. The optimizer is an algorithm which updates weights or biases to decrease the overall loss in the network with each iteration [15]. To be able to reach as close as possible to the desired output is the goal of every machine learning algorithm. We will discuss some of the optimizers and loss function that we have used to tune our Neural Networks.

4.3.1 Stochastic Gradient Descent (SGD)

The simplest yet most popular optimizer, Gradient Descent, performs well with most CNN architectures. The downside is that it requires significantly more computations when working with Big Data, as Gradient Descent computes the derivatives for each point in the sample. Stochastic Gradient Descent solves this problem. It randomly selects a point from the sample and performs the computations with only that point. SGD is very useful when there is redundant data, as it can just select one from a cluster of the sample. Since we do have many images of the same character to train the CNN, SGD performs very well here.

4.3.2 Adaptive Moment Estimation (Adam)

Adam optimizer is a unique algorithm that takes momentum into account. It adds the values of past gradients in computations, which means it can take different sized steps for different parameters and due to the momentum (past gradients accounted for) it can converge to the optimal value more quickly.

4.4 Loss Functions

Loss Functions, often called Cost Functions, are mathematical equations that essentially associates a value or event with a real number. In terms of machine learning, it maps a very complex network with many parameters down to a single value whose magnitude will give us an idea of the performance of the network. If a point in a sample does not reach the desired value, the loss functions assigns a penalty to it, meaning the weight or bias of that point will become negligible in upcoming iterations so it does not affect the network further. As our research requires a lot of classification, some of the important functions we used are Categorical Cross entropy, Mean Squared Error and Hinge Loss.

4.5 VGG-19

VGG (Visual Geometry Group) is a deep CNN used to classify images. VGG-19 is a VGG model which consists of 19 layers [16]. The layers consist of 16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer. The layers are designed such that bathes of convolutional layers are followed by a MaxPool layer. At the end are the 3 Fully Connected layers followed by the Softmax layer. The default input channel for VGG-19 is "RGB" but the model was fine tuned to take "gray-scale" images as input. Gray-scale images are significantly smaller than RGB images which made it much easier for our model to process them. VGG-19 works well when the data set is large. It can then learn for quite a few epochs and output a high accuracy. VGG-19 contains dropout layers which allows us to omit the use of matrix regularization functions. It prevents over-fitting of data which makes it a suitable CNN for our purpose.

VGG-19 was the first CNN we used on our data set. We tuned the model to take 100x100 images as input instead of the default 224x224 to lower execution time. The highest accuracy we achieved was 47.4% with SGD optimizer, categorical cross entropy and a learning rate of 0.0001. We interchanged the optimizer and loss functions to see if the accuracy could be increased but the effect was negligible.

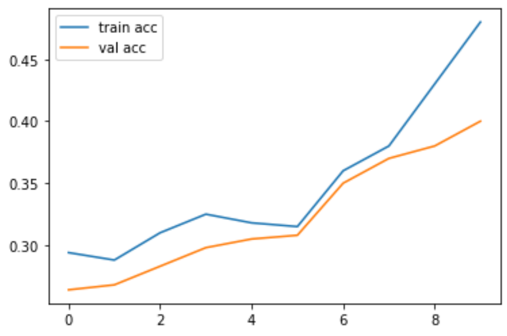


Figure 4.2: Training Accuracy graph of VGG-19

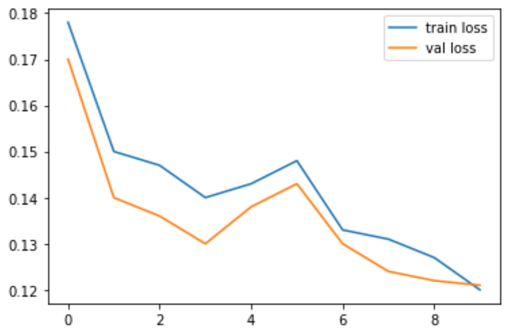
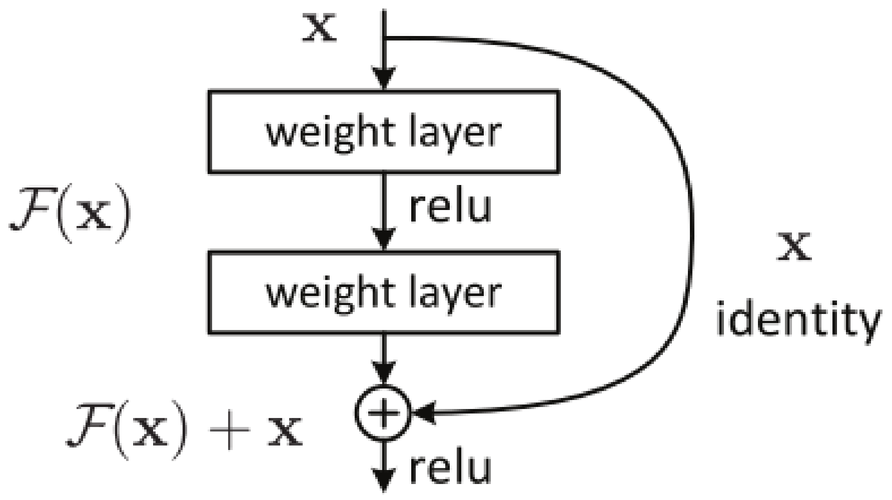


Figure 4.3: Training Loss graph of VGG-19

4.6 ResNet-50

ResNet (Residual Network) is a deep CNN used for many computer vision tasks [14]. The ResNet-50 model has 50 layers. This model has five stages each with a convolution block (each block has 3 convolution layers) and Identity block (each block also has 3 convolution layers). The ResNet-50 has over 23 million trainable parameters. It is significantly more powerful than VGG and it can detect most complex shapes and patterns. ResNet uses BatchNormalization to account for over-fitting. ResNet-50 is also fine tuned to meet our input size and dimensions.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4.4: An identity block inside ResNet.

We tuned the ResNet-50 model to take the same input dimensions as VGG-19. ResNet-50 gave a much higher accuracy of 70.3%. This was achieved using SGD optimizer, the categorical cross entropy loss function and a learning rate of 0.0001.

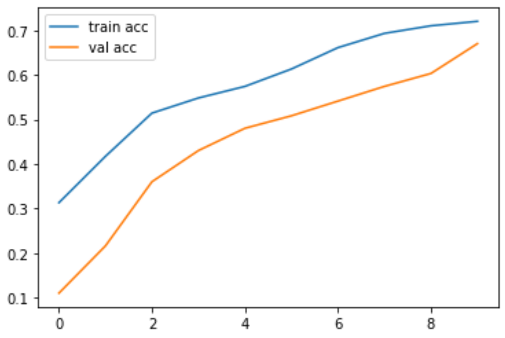


Figure 4.5: Training Accuracy graph of ResNet-50

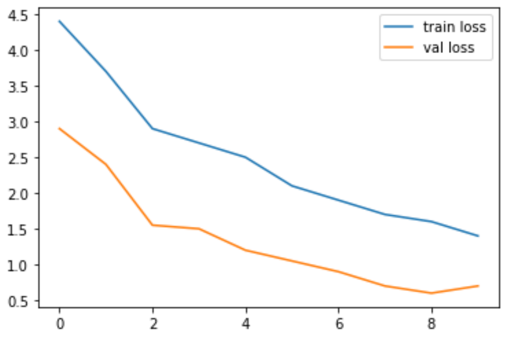


Figure 4.6: Training Loss graph of ResNet-50

4.7 Long Short Term Memory (LSTM)

Long Short Term Memory networks are a modified version of a Recurring Neural Network (RNN) [24]. Hidden LSTM layers increase the depth of the network and in turn also increases accuracy. An LSTM layer takes a 3D input and after processing it gives a 2D array as an output of the whole 3D sequence. Stacking LSTM layers in a neural network allows the network to join together previous representations and create new and more complex representation as we go deeper into the layers.

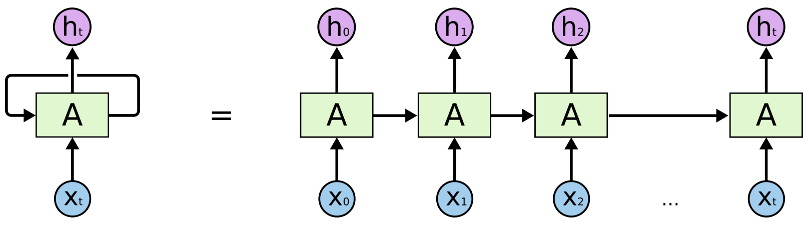


Figure 4.7: A basic sequential RNN structure.

As the model VGG-19 had the largest scope of improvement, we added 3 LSTM layers within the network. It increased the accuracy of VGG-19 up to 67.3%. Unfortunately, LSTM being a hybrid of an RNN and as it executes processing sequentially, it requires a lot of computational power. Hence, it was not feasible to add more layers or use it in conjunction with an already deep CNN.

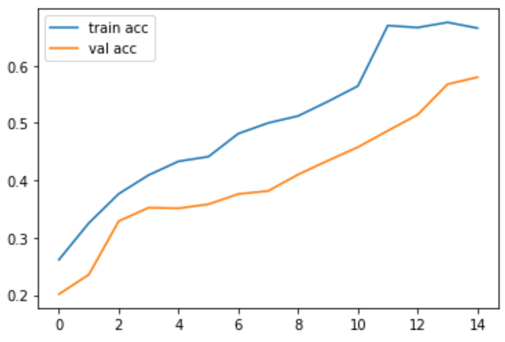


Figure 4.8: Training and Validation Accuracy graph of LSTM with VGG-19

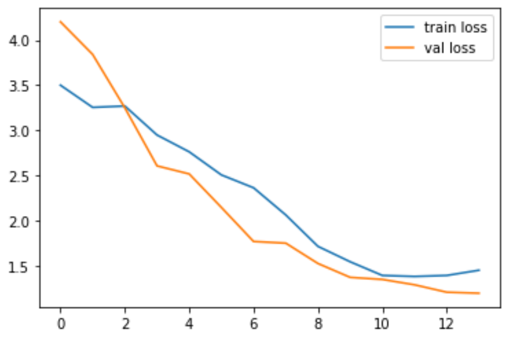


Figure 4.9: Training and Validation Loss graph of LSTM with VGG-19

4.8 GoogleNet

GoogleNet, also known as Inception V1, was proposed by Google and it won the ILSVRC 2014 with the runner up being VGG. This CNN is unique compared to others because it uses techniques such as global average pooling and 1x1 Convolutions which significantly reduce error rates. A 1x1 convolution as opposed to a 5x5 convolution needs to perform significantly less amount of operations. The CNN is composed of 22 layers and the Inception Module is built around the idea to use less computational power during execution of convolutions [17].

This model gave us an accuracy of 73.3%. Although the CNN is computationally cheap to execute, it did not perform as well as we had expected it to. The model was ran again by changing the learning rate to 0.0001 from 0.001 but the results were negligible. We suspect that this model did not perform well because for each iteration, the model was trying to set up a GPU environment as tensorflow

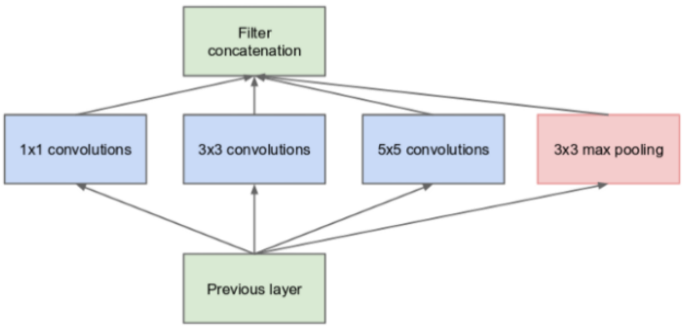


Figure 4.10: A base level Inception block.

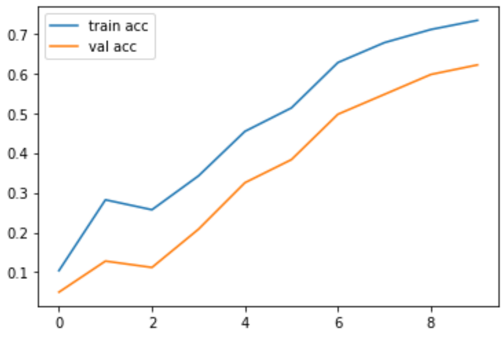


Figure 4.11: Training and Validation Accuracy graph of GoogleNet

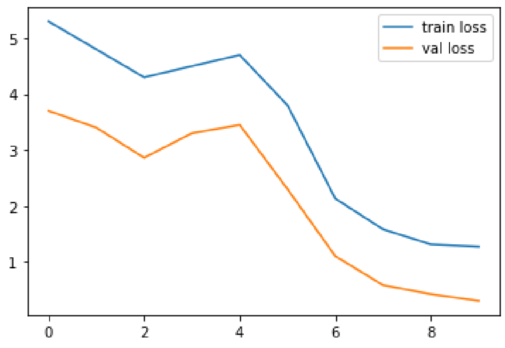


Figure 4.12: Training and Validation Loss graph of GoogleNet

libraries tend to perform better in it. Since we ran all our models in a CPU based environment, the model may have faced learning errors while processing the high quality images.

4.9 ShuffleNet

This is another CNN which is extremely efficient. Similar to AlexNet and ResNet, it uses group convolutions, but in the convolution layers it lets the layer obtain input data from different groups [15]. This allows the input and output channels to be completely related. If a convolutional layer with N groups and NxM channels of output, the output channels are reshaped into NxM dimensions, transposed and finally flattened into the next layer's input. Figure 4.22 shows a ShuffleNet Unit which consists of a channel shuffle layer between convolutions.

Initially, we ran ShuffleNet for in initial 10 epochs. The accuracy was about 71% but it was still rising. The optimizer used was Adam with a learning rate of 0.0001 and categorical crossentropy as the loss function. We then ran ShuffleNet for 5 more epochs to see how far the network can reach. But seeing the graph it appeared that the network was somewhat under fitting the data as the validation accuracy was fluctuating immensely.

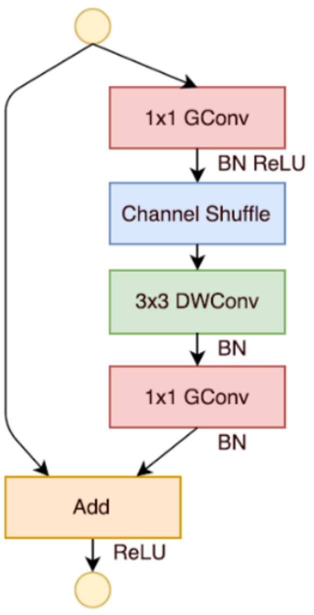


Figure 4.13: A ShuffleNet unit.

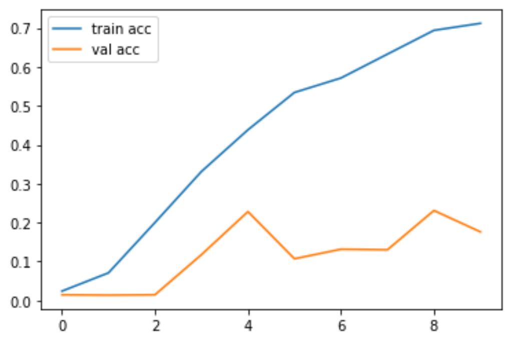


Figure 4.14: Training and Validation Accuracy graph of ShuffleNet

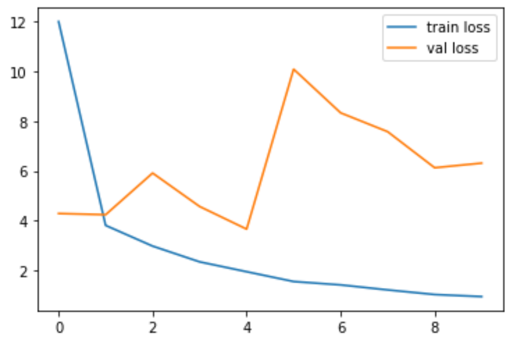


Figure 4.15: Training and Validation Loss graph of ShuffleNet

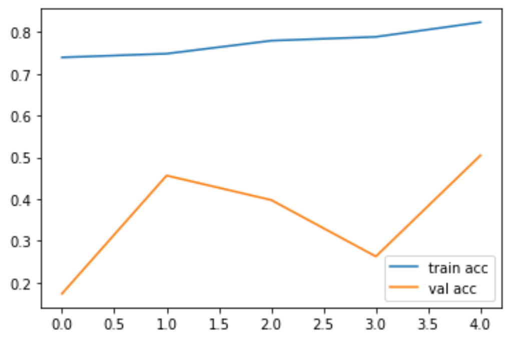


Figure 4.16: Training and Validation Accuracy graph of ShuffleNet

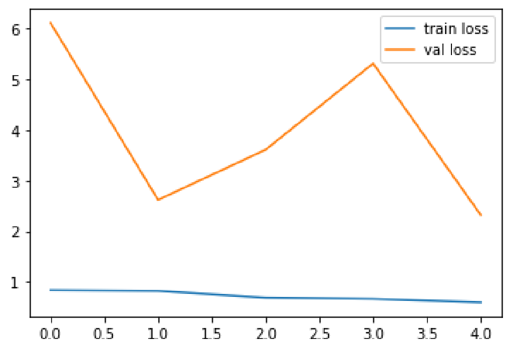


Figure 4.17: Training and Validation Loss graph of ShuffleNet

To solve this, we separated the data manually and made sure that the validation set was separated to those used during training the model. We also changed the learning rate to 0.001 from 0.0001. Since the learning rate is multiplied at the end before computing the new weights, we saw that the accuracy increase in each epoch was very small and therefore we chose to raise the learning rate.

We achieved an accuracy of about 91.2% and a significantly small classification error. The losses were also greatly reduced giving us a model that performs very well overall. As our most robust CNN, we expect it to perform even better on devices with higher computational capacity.

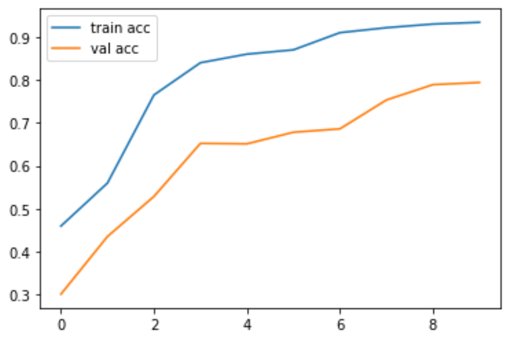


Figure 4.18: Training and Validation Accuracy graph of ShuffleNet

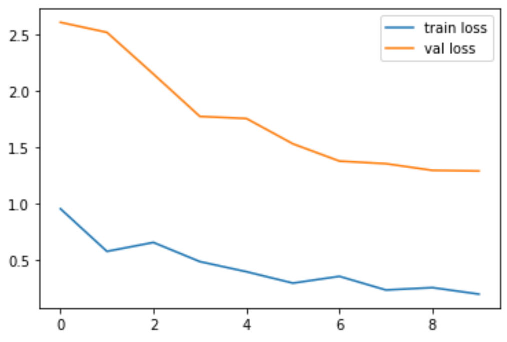


Figure 4.19: Training and Validation Loss graph of ShuffleNet

Chapter 5

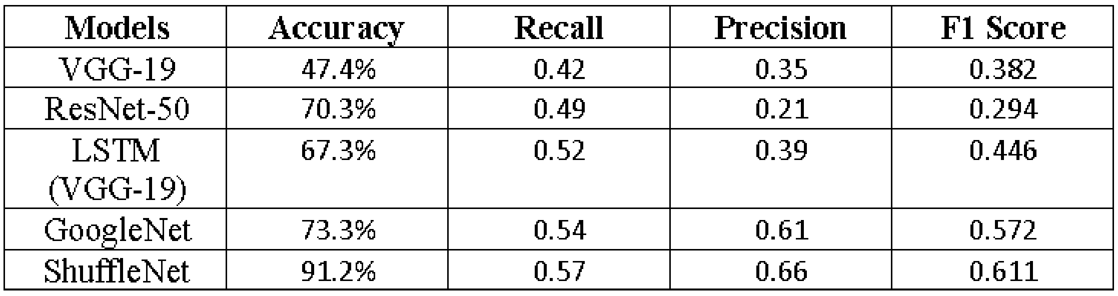
Result and Analysis

5.1 Results

After executing all the models, to quantize the performance of the models, we calculated the precision, sensitivity(Recall) and F1 scores for them. The F1 score is a weighted average of sensitivity and precision and it is considered to be a good measure for performance even if the data is skewed. The equation to calculate the F1 score is as follows:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Using BanglaLekha Isolated [7] to train our models and our own collected data for test and validation, the ShuffleNet performed the most accurately, with acceptable values of loss.



Models	Accuracy	Recall	Precision	F1 Score
VGG-19	47.4%	0.42	0.35	0.382
ResNet-50	70.3%	0.49	0.21	0.294
LSTM (VGG-19)	67.3%	0.52	0.39	0.446
GoogleNet	73.3%	0.54	0.61	0.572
ShuffleNet	91.2%	0.57	0.66	0.611

Table 5.1: A summary of the Accuracy and F1 scores

5.2 Analysis

The parameters in a network are one of the biggest factors that affect it during computation and training. All type of layers except pooling layers have trainable parameters. ShuffleNet has the least number of parameters from our models and thus it was able to perform better than the rest of the models. The parameters of a convolution layer is calculated using the following equation:

$$Parameters(Conv2D) = ((m * n * d) + 1) * k)$$

where m x n are the dimensions of the filter, d is the number of the filters in the previous layer, k is the number of filters in the current layer and 1 is added for the bias term of each filter.

CNN Model	**Number of Layers**	**Parameters (in millions)**
VGG-19	19	138
ResNet-50	50	25
LSTM (VGG-19)	23	138
GoogleNet	22	7
ShuffleNet	50	5.4

Table 5.2: A summary of the CNN models.

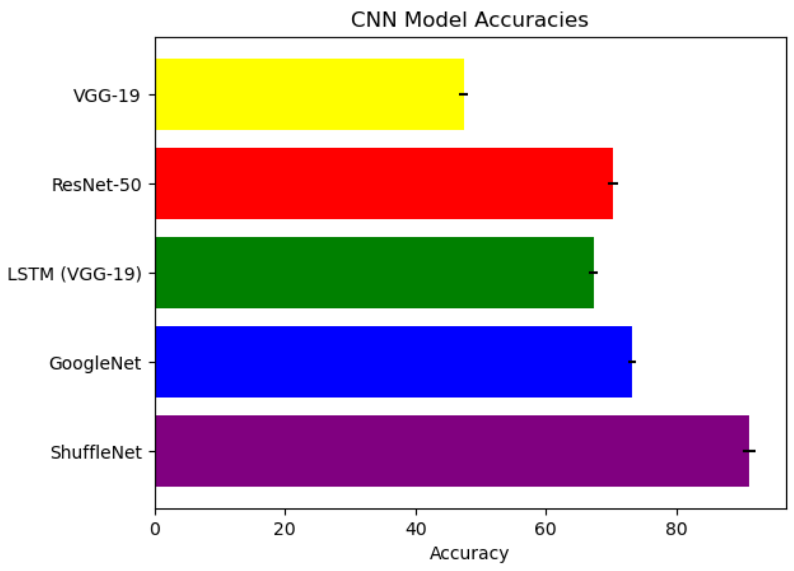


Figure 5.1: A histogram of the model accuracy rates.

Chapter 6

Future Works and Conclusion

6.1 Future Work Plan

At present the system recognizes isolated special characters with 91.2% accuracy. The results are encouraging but there is scope for improvement. The accuracy can be further improved by changing hyper-parameters such as the learning rate and batch size. We can train models on large amounts of data using a large batch size, but a higher batch size requires greater computation power that we are hoping to acquire. The robustness can be increased by running the system through larger and more extensive data. A significant improvement for the system would be to give the output in a digital text format. Several segmentation techniques could be a prominent integration to this system for better recognition of the characters. Line, word and further character segmentation of a handwritten document become inadequate when working with complex conjunct characters. Extensive horizontal and vertical segmentation are needed to be involved to breakdown the conjunct characters to their constituents. All these requires advanced computational and comprehensive skills which we aim to obtain and carry out in future. We also hope to implement our model as a web system so that institutions do not need expensive hardware upgrades to be able to use our system.

6.2 Conclusion

There have been many insightful works of handwritten text recognition over the decades and many highly accurate systems have been constructed for many languages especially in regards to English language. There have been a few fairly accurate handwritten text recognition systems for Bengali as well, but the foray into handwriting character recognition research in terms of Bengali language is still at its infancy. A successful implementation of Bengali handwriting recognition and text conversion will help reduce the workload of many and will be a big technological advancement for a developing country like Bangladesh. We hope to add to existing research and breach this gap to create a system that will be helpful for future ventures.

Bibliography

[1] U. Bhattacharya, M. Shridhar, and S. K. Parui, "On recognition of handwritten bangla characters," in *Proceedings of the 5th Indian Conference on Computer Vision, Graphics and Image Processing*, ser. ICVGIP'06, Madurai, India: Springer-Verlag, 2006, pp. 817–828, ISBN: 3540683011. DOI: 10.1007/11949619_73. [Online]. Available: https://doi.org/10.1007/11949619_73.

[2] A. Gupta, M. Srivastava, and C. Mahanta, "Offline handwritten character recognition using neural network," *2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, pp. 102–107, 2011.

[3] S. Bag and G. Harit, "A survey on optical character recognition for bangla and devanagari scripts," *Sadhana*, vol. 38, no. 1, pp. 133–168, 2013. DOI: 10.1007/s12046-013-0121-9.

[4] P. Puneet and N. Garg, "Binarization techniques used for grey scale images," *International Journal of Computer Applications*, vol. 71, pp. 8–11, Jun. 2013. DOI: 10.5120/12320-8533.

[5] M. Rahman, M. A. H. Akhand, S. Islam, P. Shill, and M. M. Rahman, "Bangla handwritten character recognition using convolutional neural network," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 7, pp. 42–49, Jul. 2015. DOI: 10.5815/ijigsp.2015.08.05.

[6] M. Biswas, R. Islam, G. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters," *Data in Brief*, vol. 12, pp. 103–107, Jun. 2017. DOI: 10.1016/j.dib.2017.03.035.

[7] M. Biswas, R. Islam, G. K. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters," *Data in Brief*, vol. 12, pp. 103–107, 2017. DOI: 10.1016/j.dib.2017.03.035.

[8] M. Z. Alom, P. Sidike, M. Hasan, T. M. Taha, and V. K. Asari, "Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018. DOI: 10.1155/2018/6747098.

[9] S. Chowdhury, F. R. Wasee, M. S. Islam, and H. U. Zaman, "Bengali handwriting recognition and conversion to editable text," *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pp. 1–6, 2018.

[10] S. Saha, M. A. B. Khaled, M. S. Islam, N. S. Puja, and M. Hasan, "Detecting sex from handwritten examples," in *2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA)*, IEEE, 2018, pp. 1–7.

[11] S. Saha, *A comprehensive guide to convolutional neural networks-the eli5 way*, Dec. 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[12] S. M. 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)*, 2019, pp. 1–6.

[13] R. R. Chowdhury, M. S. Hossain, R. ul Islam, K. Andersson, and S. Hossain, "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.

[14] P. Dwivedi, *Understanding and coding a resnet in keras*, Mar. 2019. [Online]. Available: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33.

[15] S. Bera and V. Shrivastava, "Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification," *International Journal of Remote Sensing*, vol. 41, pp. 2664–2683, Apr. 2020. DOI: 10.1080/01431161.2019.1694725.

[16] A. Kaushik, *Understanding the vgg19 architecture*, Feb. 2020. [Online]. Available: https://iq.opengenus.org/vgg19-architecture/.

[17] B. Raj, *A simple guide to the versions of the inception network*, Jul. 2020. [Online]. Available: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

[18] S. Siddique, T. Ahmed, M. Talukder, and M. Uddin, "English to bangla machine translation using recurrent neural network," *International Journal of Future Computer and Communication*, pp. 46–51, Jun. 2020. DOI: 10.18178/ijfcc.2020.9.2.564.

[19] [Online]. Available: https://shahariarrabby.github.io/ekush/#home.

[20] [Online]. Available: https://www.isical.ac.in/~ujjwal/download/SegmentedSceneCharacter.html.

[21] *Bangla Language banglapedia*, en.banglapedia.org/index.php?title=Bangla_Language, Accessed: 2015-03-31.

[22] *Free online ocr*. [Online]. Available: http://www.i2ocr.com/.

[23] K. Roy and S. Sen, *Towards online bangla handwriting recognition*. [Online]. Available: https://www.researchgate.net/publication/268813541%5C%7D.

[24] *Understanding lstm networks*. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[25] *Vision ai | derive image insights via ml nbsp;|nbsp; cloud vision api.* [Online]. Available: https://cloud.google.com/vision.

Appendix:

অ	
উ	
ঐ	
খ	
চ	
ঞ	
ঢ	
দ	
ফ	
য	
ষ	
ঢ়	

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	
বং	

Form 1 Page 1

বং	
ব্র	
বী	
বে	
ক্ক	
গ্ন	
চ্ছ্ব	
ত্ত	
দ্ভ	
ন্স	
ল্ড	
স্ত্র	

বঃ	
বৃ	
বু	
বৈ	
ক্ট	
গ্ব	
ঞ্ছ	
ত্ত্ব	
ন্ট	
ফ্ল	
ল্ব	
স্ব	

বঁ	
বা	
বূ	
বো	
ক্ত	
ঙ্খ	
ঞ্ঝ	
ত্থ	
ন্ন	
ভ্র	
ল্ল	

ব্য	
বি	
বৃ	
বৌ	
গ্ধ	
চ্ছ্র	
ট্ট	
দ্ব	
ন্ব	
ভ্ল	
স্ত	

Form 1 Page 2

অ	
উ	
ঐ	
খ	
চ	
ঞ	
ঢ	
দ	
ফ	
য	
ষ	
ঢ়	

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 2 Page 1

বং		বঃ		বঁ		ব্য	
ব্র		বৃ		বা		বি	
বী		বু		বূ		বৃ	
বে		বৈ		বো		বৌ	
ক্ক		ক্ম		ক্র		গ্ম	
গ্ল		ঘ্ন		জ্জ		জ্জ্ব	
জ্ঝ		ট্ব		ণ্ট		ণ্ঠ	
ত্ন		ত্ব		ত্ম		ন্ড	
ন্ত		ন্ত্ব		প্ট		প্ত	
প্ন		ম্ন		ম্ফ		ম্ব	
শ্চ		ষ্ক		ষ্ট		হ্ন	
হ্ম		হ্ণ					

Form 2 Page 2

অ	
উ	
ঐ	
খ	
চ	
ঞ	
ঢ	
দ	
ফ	
য	
ষ	
ঢ়	

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	
বং	

Form 3 Page 1

বং	
ব্র	
বী	
বে	
ক্ল	
ঙ্গ	
ঞ্চ	
ত্র	
ন্দ	
ক্স	
ষ্ঠ	
ব্জ	

বঃ	
বৃ	
বু	
বৈ	
স্ফ	
ঙ্ঘ	
ণ্ড	
দ্দ	
ন্ধ	
স্ল	
ষ্ক	
ব্দ	

বঁ	
বা	
বূ	
বো	
ক্ম	
জ্ঞ	
ণ্ন	
দ্ধ	
প্প	
ল্ক	
স্ক্র	

ব্য	
বি	
বৃ	
বৌ	
ঙ্ক	
জ্ব	
ণ্ম	
ন্ত্র	
প্ল	
ল্ট	
হ্ল	