Bangla Isolated Handwritten Basic Characters, Compound Characters and Digits Recognition Using Deep Convolutional Neural Network

Chandrika Saha
Registration No.: 13CSE014
Session: 2013-2014

A Thesis Presented for The Degree of
Bachelor of Science

Department of Computer Science and Engineering
University of Barishal
Bangladesh
19-03-2019

This research work is dedicated to my family.

Bangla Isolated Handwritten Basic Characters, Compound Characters and Digits Recognition Using Deep Convolutional Neural Network

Approved:

Supervisor	Date
Student	Date

Acknowledgements

I would like to take this occasion to thank God, almighty for taking my endeavor to a successful culmination. I extend my sincere and heartfelt thanks to my esteemed supervisor, **Md. Mostafijur Rahman** for providing with the right guidance and advice at each steps of this work. I would like to thank the other faculty members also for supporting and encouraging this work. I would also like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Publications

- "Bangla Handwritten Character Recognition using Local Binary Pattern and its Variants," in International Conference on Innovations in Science, Engineering and Technology, 2018. (Accepted and presented)
- "Bangla Handwritten Digit Recognition using an Improved Deep Convolutional Neural Network Architecture," in International Conference on Electrical, Computer and Communication Engineering, 2019.(Published)
- "Bangla Handwritten Basic Character Recognition Using Deep Convolutional Neural Network", in Joint International Conference on Informatics, Electronics and Vision (ICIEV) and International Conference on Imaging, Vision and Pattern Recognition (IVPR), 2019. (Accepted)

Contents

List of Figures

List of Tables

Abstract

Bangla is one of the mostly spoken languages all over the world. Optical Character Recognition (OCR) is a vital task for any language because of its use in different sectors of life. Handwritten Character Recognition (HCR) is a form of OCR that classifies human written texts. HCR is a very useful tool for modern era of digitization. We need HCR for data digitizing, robotics vision, helping visually disabled people and many more sectors. For building a complete HCR, first the text component from an image need to be located, then sentences and words and lastly isolated characters are needed to be separated. So, it is needed to ensure that we have a classifier that can classify isolated characters excellently. Isolated handwritten character recognition is considered a hard image classification problem as writing style varies human to human. However, despite Bangla being a popular language, there is relatively less number of research works on Bangla HCR. This thesis aims to find an effective and efficient technique to recognize Bangla isolated characters and digits. After exploring several state-of-arts machine learning technique it was found that a better approch is needed for such a difficult task. Now-a-days, Deep Convolutional Neural Network (DCNN) is gaining popularity for better performances for image classification but in the case of Bangla handwritten characters it is relatively less explored. So, a DCNN model for Bangla handwritten basic characters, compound characters and digits recognition is proposed. Rigorous experimentation on Bangla handwritten datasets have showed superior accuracies than state-of-arts machine learning algorithms. Using a 14 layered DCNN model containing 6 convolutional layer, 6 max pooling layer and 2 fully connected layer with dropout regularization, namely BanglaNet-14 a recognition accuracy of 95.62% is obtained for a total of 231 classes of Bangla basic, compound characters and digits. For evaluation three benchmark datasets namely CMATERdb 3.1.2 (basic characters), CMATERdb 3.1.3.1 (compound characters) and CMATERdb 3.1.1 (digits) are used. For CMATERdb

3.1.2, the proposed improved DCNN model gives a recognition accuracy of 97.67% , for CMATERdb 3.1.3.1 a recognition accuracy of 96.33% and for CMATERdb 3.1.1 accuracy of 98.35% is obtained. For building a complete Bangla HCR we need to have a classifire that can detect any Bangla basic characters, compound characters and digits. For this purpose CMATERdb 3.1.2, CMATERdb 3.1.3.1 and CMATERdb 3.1.1 datasets were combined together that forms a total of 231 classes. Recognition accuracy for this dataset is 95.62%. Also two differnt datasets, namely, BanglaLekha isolated and Ekush dataset containing 84 and 122 classes respectively are also considered for evaluating handwritten basic, compound characters and digits. The improved DCNN model yields a recognition accuracy of 95.94% on BanglaLekha isolated dataset and 93.86% accuracy on Ekush dataset. All these experimental evaluation is shown to give superior recognition accuracies then existing prominent techniques.

Chapter 1

Introduction

1.1 Overview and Objectives

Optical Character Recognition (OCR) is the conversion of images of characters into electrical format. The electrical form of text is a better way of storing, searching, displaying, editing and translating data. OCR is used extensively in machine learning, natural language processing, robotics vision, reasoning, text to speech conversion and many other sectors. We see the use of OCR in checking business documents, number plate recognition, automatic key information extraction, pen computing and helping visually impaired people [1]. Handwritten Character Recognition (HCR) is a special type of OCR which deals with the ability of computer to recognize the letters written by human. HCR has become a vital part of robotics vision for its application in enormous sectors of life like smart education, autonomous driving, purchasing process, digitizing paper based process and many other areas [2]. Actually, we find the touch of education around us every day, everywhere.

Bangla is the fifth largest spoken language around the world being the language of 242 millions of people. It is the mother tongue of people of Bangladesh and also official languages of some Indian states [3]. Bangla has a rich set of characters including 11 vowel, 39 consonants, 171 compound characters and 10 digits. Compound characters can contain 2 or 3 basic characters combined into a single character. For classifying Bangla handwritten characters, we need to use some feature extraction and classification method. There are a variety of feature extraction techniques that have been used for Bangla HCR like shadow and quad tree based features [4, 5, 6], gabor filters, zoning [7], gradient based features[8, 9] etc. Recently, Local Binary

Pattern (LBP) based feature descriptors [10, 11, 12, 13, 14, 15] have become popular for their robustness, simplicity and considerable accuracies. Already Hassan et al. have proposed a method for Bangla digit recognition using LBP as feature descriptor [16] which has obtained better results. However, works using LBP based feature descriptors are hardly seen in Bangla HCR. Some classification algorithms like logistic regresssion, neural network, Support Vector Machine (SVM) [17] etc. uses the features obtained by any feature extraction method to classify Bangla characters. The vital part of categorizing characters is feature extraction as it brings out the core part from a given data. The main concern of this paper is to apply various type of LBP based new feature descriptors on Bangla handwritten characters and find out which method works superior.

Again, Deep learning is a tool that has enhanced the state of solution for many problems. It is taking over because with more data deep learning keeps doing better which state of art machine learning techniques failed to do in most cases. Deep Convolutional Neural Network (DCNN) has shown better results for image classification tasks as it can detect infinitesimal details from an image. The shallower layer of a DCNN can detect more visible features of an image like color and edges, deeper layers can detect more complicated features. Thus, DCNN performs a learning process that goes from surface to in depth to recognize something which is somewhat like the learning process of human [18]. Handwritten Character Recognition (HCR) is one of the hardest classification problems. Encompassing DCNN for HCR can improve the recognition accuracy because of its learning process [19]. first and foremost objective of this thesis is to find a effective and superior way to classify Bangla isolated characters and digits so that main task for building a complete Bangla HCR is achieved. Other objectives of this thesis includes:

- Exploring state-of-art machine learning techniques for Bangla handwritten character and digit classification to find out their effectiveness. Using LBP based feature descriptors on Bangla isolated characters and digits.
- Exploring DCNN to understand and use it for Bangla isolated character and digit classification. Applying LBP based feature descriptors on Bangla isolated characters and digits.
- Building a efficient, effective and improved DCNN model to en-

sure performance gain over existing methods for benchmark datasets.

1.2 Motivation

As mentioned earlier, now-a-days we cannot do a thing without education. The base of smart education is handwritten character recognition. Manisha et al. in their research paper discussed various applications of the offline handwritten character recognition [6]. Again, Bangla is one of the mostly spoken languages of the world but handwritten character recognition technique for Bangla language is not that developed. As Bangla is our mother tongue we should feel the need of building a handwritten character recognition tool for Bangla language. Following are some of the many sectors where we can use handwritten text recognition technique:

- **Digital Character Conversion:** Handwritten character recognition can make the data readable to robotics agents. Also it can be used to convert non readable format of data to readable format.
- **Meaning Translation:** Handwritten characters can be used to form meaningful words by which meaningful sentences can be created. These sentences can then be converted to another language. Stories, literatures even lecture notes can be converted to other languages so that people from different countries can understand them.
- **Content Based Image Retrieval:** Content based image retrieval based on text, in which the user enters a word in the search box; the search engine then retrieves the document images that contain the search word.
- **Keyword Spotting:** Given a sample image mostly repeated word from that image can be retrieved which we can use as a keyword.
- **Signboard Translation:** Signboard Translation can be very effective thing for travelers. Typically, signboards are written in the native language of any country which travelers cannot decode. Using signboard translation one can take a picture of the signboard, get the electrical form of the text and translate it to an appropriate language.

- **Text-to-Speech Conversion:** Converting the text of any handwritten document image into speech is called text-to-speech conversion. This can be used in various sectors like; blind students can use this to listen words from text books.

- **Scene Image Analysis:** Characters can be identified among many different objects of a scene. This can be used for identifying general purpose object detection.

1.3 Challenges

Isolated handwritten character classification is a challenging classification task. The challenging aspects of isolated handwritten character classification is as follows:

- **Different Style of Writing:** Although all characters and digits are supposed to be similar in shape but different human have different way of writing the same characters. Bangla has a rich set of characters and digits almost all of which are cursive in nature. As a result, this variation is very noticeable in the case of Bangla dataset. This is one of the biggest challenging aspect of any isolated handwritten character classifier.

- **Use of Compound Characters:** Compound characters are two or more basic characters combined together. There are almost 300 types of possible compound characters. Such huge number of complex shaped characters made the task for classification much harder.

- **Cursive Nature of Characters:** The cursive way of writing Bangla characters and digits made it easier to confuse between characters and making similar looking but different characters. Sometimes it is hard for even human to recognize writing of other individuals.

- **Different Way of Writing Same Characters:** There are quite a number of characters having more than one form of writing. Such characters make the task of classifier very difficult as for different shapes of characters it needs to classify them in same category.

1.4 Research Questions

Considering the aforementioned issues, the objective of this research is to design a efficient technique for Bangla isolated basic characters, compound characters and digits. In this research, this following research question will be answered,

How to design a efficient technique to classify Bangla handwritten basic characters, compound characters and digits? More specifically, How to improve the performance of classification task of isolated Bangla basic, compound characters and digits?

1.5 Thesis Organization

The rest of this thesis is organized as follows.

- Chapter 2: This chapter discusses recent significant works on Bangla isolated basic characters, compound characters and digits recognition.
- Chapter 3:This chapter discusses the use of some state-of-arts machine learning techniques to find out their effectiveness on the current problem.
- Chapter 4: A detailed overview of terminologies and functioning of DCNN is given in this chapter along with the proposed improved DCNN model.
- Chapter 5: This chapters gives a detailed description of the results obtained from applying the improved DCNN model along with different datasets used for experimental evaluation.
- Chapter 6: This chapter concludes the thesis with future guidance.

Chapter 2

Literature Review

2.1 Introduction

In this Chapter recent significant works for Bangla basic characters, compound characters and digits are discussed. Section 2.2 discusses the works on Bangla isolated basic characters, Section 2.3 discussed research works of handwritten compound characters and Section 2.4 described works on handwritten digits.

2.2 Bangla Handwritten Basic Characters

There have been some works on Bangla basic characters, most of which are state of arts machine learning algorithm based. Some of these research works are briefly described in this section.

Bhowmik et al. proposed a recognition scheme for Bangla basic characters based on MLP classifier [20]. Stroke features extracted from the images were used as feature set for the classifier. Stroke features are basically vertical and horizontal edge like features extracted from a binary image. Such vertical and horizontal strokes are used to recognize shape of particular character. For evaluation they used a customized dataset that includes a total of 25000 samples of 50 basic characters. They further extended their study by performing a comparative study of Bangla basic character recognition using MLP, SVM and radial basis function network with wavelet features as feature descriptor [21]. This time a dataset with 27000 samples were used for 45 classes as characters with strong structural similarities were considered same classes. Basu et al. introduced a MLP based classification technique for Bangla basic character [4]. Centroid, shadow and the longest run based features were used as feature set. A dataset of 10000

image samples of 50 basic characters was formed for evaluation. Das et al. proposed an enhanced feature extraction scheme in their study where MLP with one hidden layer was used as classifier [5]. They used 8000 characters for training the model and 2000 characters for testing. In another study, Pervin et al. proposed a method that uses fusion of zoning and gabor filter as feature descriptor [7]. They have used a benchmark dataset named Banglalekha isolate [22] for evaluating the proposed scheme. For training and testing 500 and 200 images respectively were considered for each classes. In [23] different variations of LBP based feature descriptors were used for classifying basic, compound characters and digits. For classification SVM with linear kernel was used. For evaluating performance on basic character recognition, CMATERdb 3.1.2 [4, 5] dataset was used. In another study, Rahman et al. proposed a DCNN based scheme for recognizing Bangla basic characters [24]. They used a database of 20000 images for training and testing the model. Their proposed model is basically a 6 layered CNN containing 2 convolutions, 2 pooling, 1 fully connected and 1 output layer. Rabby et al. proposed a CNN based model for Bangla HCR named EkushNet [25]. They have used a total of 368776 images for 50 basic characters, 10 modifiers, 10 digits and 52 mostly used compound characters.

2.3 Bangla Handwritten Compound Characters

There have been several research works on Bangla handwritten compound character classification which includes a diverse range of techniques for classification.

Pal et al. proposed a gradient feature based method for classifying offline compound characters [8]. On 20,543 images of 138 chosen compound character classes they applied a five-fold cross validation based classifier. As feature descriptor they used direction information achieved from gradient based data. Das et al. proposed a scheme where 43 mostly used compound characters were considered [26]. As feature descriptor they used a quad tree based feature set. As classification technique they used Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM) based classifier. In [23], recognition scheme for Bangla handwritten basic, compound characters and digits was proposed. For experimentation, CMATERdb dataset was used. As classifier, SVM with linear kernel was applied. Local Binary Pat-

tern (LBP) based feature extraction methods [10, 12, 13, 14, 27, 11, 15, 28] were used for this research. Roy et al. proposed a Supervised Layer wise training of a Deep Convolutional Neural Network (SL-DCNN) model that takes an incremental approach to train a model [29]. They kept increasing the number of convolution and pooling layer and train the model before adding more layers. On CMATERdb 3.1.3.3 dataset [30] the evaluated their model with RMSProp as optimization technique. Ashiquzzaman et al. proposed a Deep Convolutional Neural Network (DCNN) based method where they used Exponential Linear Unit (ELU) as activation unit and dropout regularization [31]. They teste and trained their model on CMATERdb 3.1.3.3 dataset with 171 classes. Keserwani et al. proposed a two phased DCNN scheme for Bangla compound character recognition [32]. Instead of randomly initializing weights of DCNN, they first trained a CNN to minimize the reconstruction loss and used weights gained from this phase to initialize weights of phase two. Adadelta optimization technique and ReLU activation were used while training the model. Pramanik et al. proposed a shape decomposition based scheme where the compound characters were segmented into basic characters and then they were classified [33]. MLP classifier with backpropagation is used as classification technique with chain code histogram feature descriptor. The images were preprocessed using granular noise cleaning and binarization. Sharif et al. proposed a CNN based model that combines Histogram of Oriented Gradients (HOG) features with the features of CNN [34]. To train the model they used Adadelta optimizer. They evaluated their proposed model using CMATERdb 3.1.3.3 dataset with 171 classes and 199 classes separately. They used data augmentation by retaining the original image. As part of preprocessing they resized images to a certain size and changing the background of the character to black and the character itself to white color.

2.4 Bangla Handwritten Digits

The research works on Bangla OCR began at 1980s [7]. However, most of the works since then was printed document based. Works on Bangla HCR are relatively few. In this Section some remarkable works on Bangla handwritten digit recognition are discussed.

An automatic offline Bangla handwritten numeral recognition method was proposed by Pal et al.[35]. A model based on features obtained

from the concept of water overflow was used in this study. The region of the character is imagined to be a water reservoir where the direction of water overflow from the reservoir, position of the reservoir according to the character bounding box, height of water level, shape of reservoir were used as recognition scheme. On collected data from varying individuals they obtained 91.98% recognition accuracy. Pal et al. further extended their studies on Bangla handwritten numerals using the same types of features obtained before [36]. On a dataset of 12000 images they obtained 92.8% accuracy. Khan et al. proposed a scheme for Bangla handwritten character recognition that uses a sparse representation classifier based method[37]. They obtained 94% accuracy on CMATERdb3.1.1 dataset [38, 39]. They used zone density as feature extraction method. Another study was proposed by Hassan et al.[16] where they used Local Binary Pattern (LBP)[10] as feature descriptor and K-Nearest-Neighbor (KNN) algorithm for classification. They obtained 96.7% accuracy on CMATERdb 3.1.1 dataset. Aziz et al. introduced a method where they used local gradient direction pattern based feature descriptor [9]. For classification they used KNN and SVM. On CMATERdb3.1.1 dataset 95.62% accuracy was obtained. Another study [23] suggested a Bangla handwritten character recognition scheme using Local Binary Pattern (LBP) [10] based feature descriptors [14, 12, 13, 27] and Support Vector Machine based classification for Bangla basic characters, compound characters and digits. Very recently, alom et al. has developed a five layered D-CNN based architecture for Bangla digit recognition [40]. They evaluated the performance of CMATERdb 3.1.1 dataset using D-CNN with various filters and dropouts.

2.5 Summary

From the aforementioned discussion we can see that most of the researches conducted on Bangla HCR uses quad tree, centroid, shadow based features, gabor filter, gradient direction based approaches and rarely concentrated on the exploration of LBP based new feature descriptors. Therefore, further studies need to be conducted on Bangla HCR to address the performance of new LBP based feature descriptors on it. Again many different types of state-of-arts classifiers like MLP, SVM, KNN, K-means are used for classification task. Even some papers proposed DCNN based approach. However, it can be seen that

effective DCNN based research works is not many in numbers. On account for this, a new enhanced DCNN model for Bangla handwritten basic characters, compound characters and digits is needed.

Chapter 3

Preliminary Study

3.1 Introduction

Before finding out a efficient DCNN model for Bangla isolated character and digits, there is need to first explore some of the state-of-art techniques in existence. A typical machine learning technique works by first extracting features from given input images, then normalizing these features and lastly using a classification algorithm to fit a function to the given problem. There are a variety of feature extraction techniques that have been used for Bangla HCR like shadow and quad tree based features [4, 5, 6], gabor filters, zoning [41], gradient based features[8, 9] etc. Recently, Local Binary Pattern (LBP) based feature descriptors [10, 11, 12, 13, 14, 15, 27, 28] have become popular for their robustness, simplicity and considerable accuracies. However, works using LBP based feature descriptors are hardly seen in Bangla HCR. So, some LBP based new feature descriptors are applied to see their performance on Bangla handwritten data.In this chapter those new LBP based feature descriptors are described along with their performances on some benchmark Bangla handwritten characters and digits datasets. This chapter describe the steps involved in the completion of a state-of-art machine learning technique.

3.2 Feature Extraction

Feature extraction is the process of capturing significant data from any given input. In this paper, seven LBP based feature descriptors are considered for feature extraction. These techniques find out specific patterns or principle components from a given image. Short description of each of the methods is given in the following Subsections.

3.2.1 Local Binary Pattern (LBP)

Original LBP operation is first explored by Ojala et al. [10]. LBP considers N x N (typically 3 x 3) neighborhood of the center pixel value and find out a binary result by comparing this value with all the neighbors. The obtained binary number is then converted into a decimal value. For calculating basic LBP (3.1) is used.

$$LBP_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(p_i - p_c)2^i, \quad q(d) = \begin{cases} 1, & \text{if } d \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Here, n is the total number of neighbors of any central pixels having radius r. The coordinates of central pixel is (x_c, y_c). p_i is the pixel value of i$^{\text{th}}$ neighboring pixel and p_c is the central pixel value. ''d" is the difference between the i$^{\text{th}}$ neighboring pixel value and central pixel value. If this difference is greater than or equal to 0 then 1 is placed in the place of i$^{\text{th}}$ neighboring pixel otherwise 0 is placed. Afterwards, a binary string is generated considering all the neighboring pixels starting from (0, 0) coordinate in a clockwise direction. This binary string is then converted into decimal value which is the final result. Fig. 3.1 gives an example of this process. LBP can detect micro-structures like corners, edges and line ends. However, main drawback of LBP is lack of illumination adaptability and noise intolerance nature.

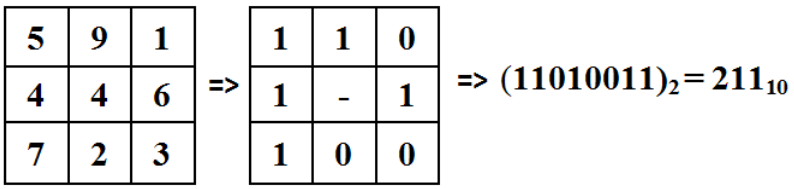


Figure 3.1: Example of LBP calculation.

3.2.2 Local Gradient Pattern (LGP)

LGP feature descriptor is proposed by Jun et al. [11] which uses gradient value of the differences of all the neighbors from the central pixel as threshold. LGP operation is defined by (5.1).

$$LGP_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(g_i - g_\mu)2^i, \quad q(d) = \begin{cases} 1, & \text{if } d \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Here, n and r is the total number of neighbors and radius respectively. (x_c, y_c) is the coordinate of the central pixel. g_i is the absolute value of

differences between central pixel intensity and ith pixel intensity and $g_\mu = \frac{1}{n}\sum_{i=0}^{n-1} g_i$. Fig. 3.2 gives an example of calculation used in LGP method.

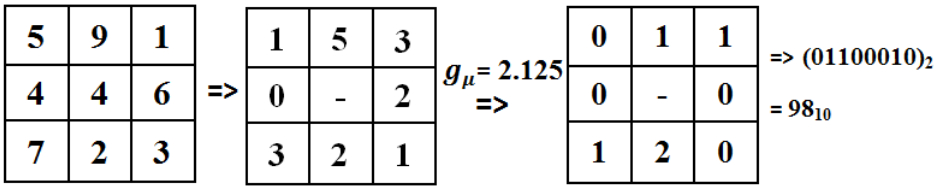


Figure 3.2: Example of LGP calculation.

3.2.3 CENsus TRansform hISTogram (CENTRIST)

CENTRIST performs a census transform (CT) of an image and replaces the image with the CT value [27]. CT is a non-parametric local transform that is designed for forming correspondence between local patches [42]. The whole process of CENTRIST is illustrated in Fig. 3.3. The only difference between LBP and CENTRIST is that interpolation of corner pixels is not used in CENTRIST.

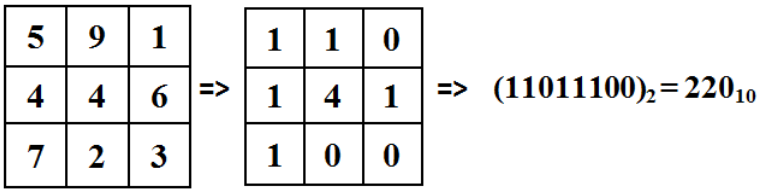


Figure 3.3: Example of CENTRIST calculation.

3.2.4 Noise Adaptive Binary Pattern (NABP)

NABP eradicates the limitations of LBP by using adaptive thresholds [14]. NABP has the property of adapting illumination variations and noise tolerance unlike LBP. For NABP calculation (3.3) is used.

$$NABP_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(p_i - p_c)2^i, \quad q(d) = \begin{cases} 1, & \text{if } d \geqslant T_a \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Here, n and r is the total number of neighbour and radius respectively. (x_c, y_c) is central pixel coordinate. T_a is the adaptive threshold which is derived taking the square root of p_c, the central pixel intensity. The overall process of NABP is described using an example on Fig. 3.4.

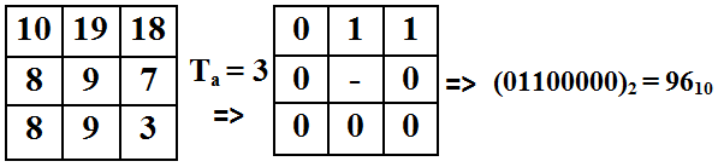


Figure 3.4: Example of NABP calculation.

3.2.5 Local Ternary Pattern (LTP)

Tan et al. [15] proposed LTP being inspired by LBP. LTP is a ternary method which uses three bits to tackle intensity variation in place of 2 bits in LBP. LTP uses two thresholds +5 and -5 for getting upper and lower pattern. LTP is defined in (3.4).

$$LTP_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(p_i - p_c)3^i, \quad q(d) = \begin{cases} +1, & \text{if } d \geqslant 5 \\ -1, & \text{if } d \leq -5 \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Here, (x_c, y_c) is the coordinate of central pixel. p_c is the intensity of central pixel and p_i is the intensity of i^{th} neighbouring pixel. The overall process of LTP is shown in Fig. 3.5.

3.2.6 Local Adaptive Image Descriptor (LAID)

LAID is a variation of LTP propoed by Ishraque et al. [28] that uses dynamic threshold instead of a static one in LTP. LAID also produces two patterns namely upper and lower pattern. Equation (3.5) defines LAID operation.

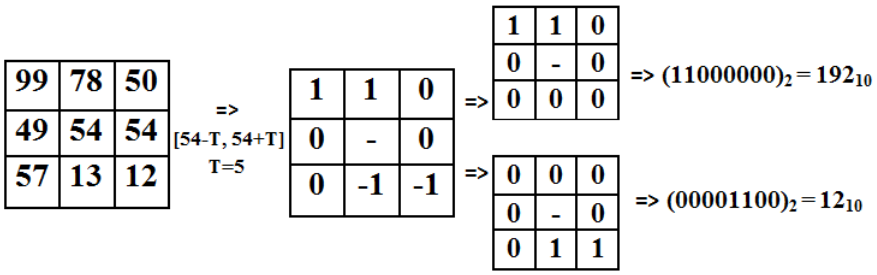


Figure 3.5: Example of LTP calculation.

$$LAID_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(p_i - p_c)3^i, \quad q(d) = \begin{cases} +1, & \text{if } d \geqslant T \\ -1, & \text{if } d \leq -T \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Here, (x_c, y_c) is the central pixel coordinates and p_c is the intensity of central pixel. Firstly, The difference among this central pixel intensity and the neighboring pixels is calculated and then, the result is compared with threshold T which is computed by taking the median of differences of all the neighboring pixels from the center pixel. The code splits the resultant matrix into upper pattern and lower patterns. Fig. 3.6 shows an example of LAID calculation.

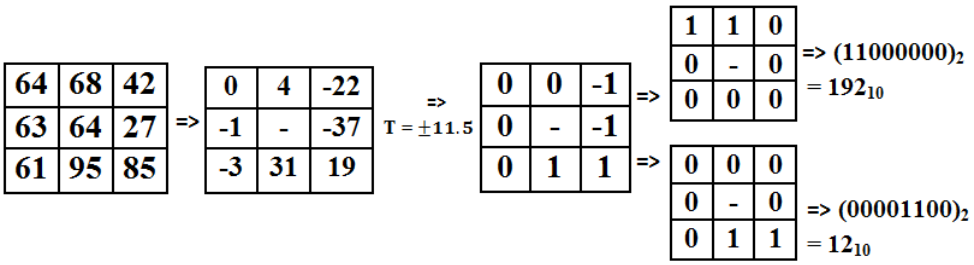


Figure 3.6: Example of LAID calculation.

3.2.7 Discriminative Ternary Census Transform Histogram (DTCTH)

DTCTH is proposed by Rahman et al. which is a ternary feature descriptor [12]. DTCTH is calculated using (3.6).

$$DTCTH_{n,\,\mathrm{r}}(x_c, y_c) = \sum_{i=0}^{n-1} q(p_i - p_c)3^i,$$
$$q(d) = \begin{cases} +1, & \text{if } d \geqslant T \\ -1, & \text{if } d \leq -T \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

Here, (x_c, y_c) is the central pixel coordinate. n, r is the number of neighboring pixels and radius respectively. DTCTH considers the square root of central pixel as threshold value, T. This choice increases the discriminative and noise adaptive nature. Like LTP, DTCTH divides the code into two patterns namely upper and lower pattern. The lower group is considered as background of the image whereas the upper group is the foreground. DTCTH calculation is depicted in Fig 3.7.

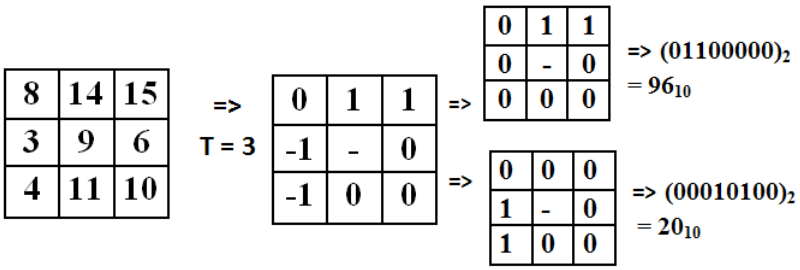


Figure 3.7: Example of DTCTH calculation.

3.3 Feature Representation

Feature representation is used to capture the global structure of an image by integrating data collected from the sub regions of an image. For this study, overlapped Spatial Pyramid Matching (SPM) scheme is adopted [42]. SPM takes histograms from sub-regions of an image to form a final histogram that represents the whole image. On this final histogram, Principle Component Analysis (PCA) algorithm is applied to select main features to be used in classification.

3.4 Feature Normalization

Feature normalization is applied to standardize the range of values of features. As the range of values of features can vary significantly, it is necessary to normalize them so that they can work properly with classification algorithms. For this study, features are normalized using (3.7) such as they have unit norm and zero mean.

$$N(x_i) = \frac{(x_i - x_\mu)}{\sqrt{\sum (x - x_\mu)^2}} \tag{3.7}$$

Here, x is the array of features, x_μ denotes the mean value of features and x_i is the i^{th} feature from the array.

3.5 Classification

Classification is the process of labeling input images into certain categories. There are many algorithms for classifying data such as logistic regression, neural network, Support Vector Machine (SVM) etc. SVM with linear kernel is used in this study as classification algorithm. SVM is a well-known supervised machine learning algorithm developed by Cortes et al. [17]. SVM is known to provide higher accuracies than

other machine learning algorithms because of its largest separation margin property.

3.6 Summary

For experimenting, three benchmark datasets namely CMATERdb 3.1.2, CMATERdb 3.1.3.1 and CMATERdb 3.1.1 for Bangla basic, compound characters and digits respectively are used. These datasets are evaluated separately by dividing them into five parts containing random images. Experiment is done for these five parts i. e, fivefold

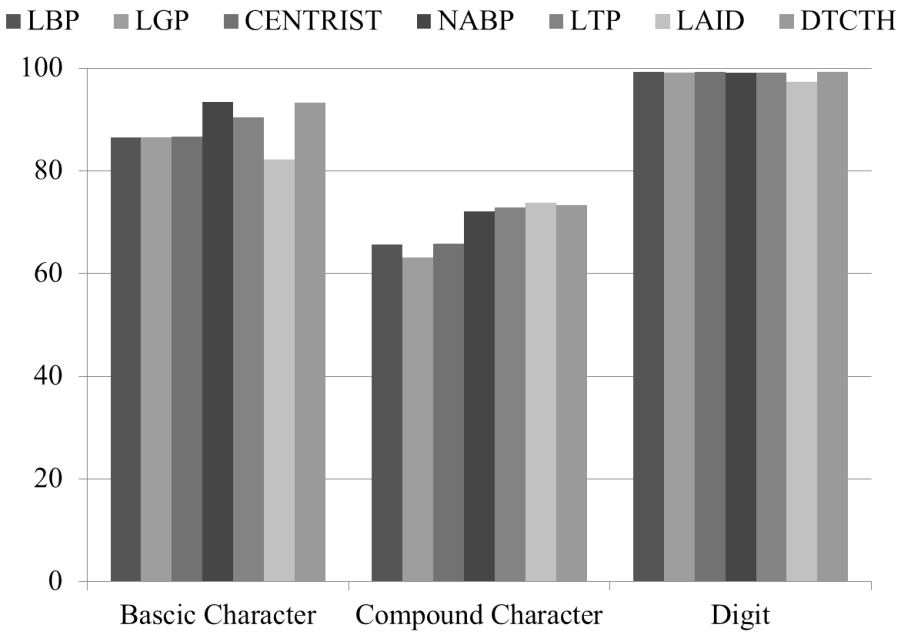


Figure 3.8: Average accuracy for Bangla basic, compound characters and digits.

cross validation is performed.For this work, CENTRIST[1] framework is adopted; keeping the parameters used in CENTRIST same. For fair comparison, the eight neighboring pixels of the central pixel are used for all the methods, though different parameter settings may produce better results. As feature representation technique a three level overlapped SPM scheme is used. Level 2 in SPM divides the image into 25 sub-regions, level 1 and 0 divides it into 5 and 1 regions respectively. Following CENTRIST, corner point interpolation is avoided and two DCT bins, 0 and 255 is removed. After applying overlapped SPM, 40 most significant features are identified using PCA. For classification,

[1]https://www.nju.edu.cn/_upload/tpl/00/ed/237/template237/publication.htm

SVM classifier is used with linear kernel where $c = 2^{-5}$ and $g = 2^{-7}$. LIBSVM[2] library is adopted for implementing SVM.

Fig. 3.8 (pg. 30) depicts a bar graph showing the average accuracies obtained in the three categories of datasets and table 3.1 gives the experimental results.

For Bangla basic character dataset (CMATERdb 3.1.2), we can see that, LBP (86.56%) and CENTRIST (86.64%) gave lower accuracies than NABP as they lack noise adaptiveness and discriminative power. Another version of LBP, namely LGP also produced similar accuracy (86.56%) with average value based adaptive threshold. On the other hand DTCTH perform much better (93.24%) than other similar ternary methods because of it's discriminative power. LTP with it's static threshold obtained 90.34% accuracy. LAID obtained 82.16% accuracy which uses adaptive threshold as mentioned earlier. Methods which uses average based thresholds like LGP and LAID obtained lower accuracuies as they can be affected by out-layer.

Table 3.1: Effect of Different Feature Descriptors on Bangla Isolated Handwritten Characters

	CMATERdb 3.1.2	**CMATERdb3.1.3.1**	**CMATERdb 3.1.1**
LBP	86.56%	65.69%	99.24%
LGP	86.56%	63.17%	99.08%
CENTRIST	86.64%	65.78%	99.24%
NABP	93.40%	72.11%	99.08%
LTP	90.34%	72.84%	99.16%
LAID	82.16%	73.78%	97.26%
DTCTH	93.24%	73.3%	99.2%

In the case of compound character dataset (CMATERdb 3.1.3.1), it can be seen that LAID and DTCTH descriptors performed much better than other methods, obtaining 73.78% and 73.3%, accuracies respectively. NABP (72.11%) performed better than basic LBP (65.69%) as considering square root of central pixel increases it's discriminative power. Considering average based threshold LGP has obtained 63.17% accuracy. CENTIRST obtained 65.78% accuracy with fixed threshold. Among the ternary feature descriptors LAID (73.78%) performed better because it can detect abrupt changes due to local illumination

[2]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

variations. Again, DTCTH with adaptive threshold performed better than basic LTP (72.11%) with fixed threshold.

As in Bangla digits dataset (CMATERdb 3.1.1), almost all the methods performed very well on this dataset with around 99% accuracies. LBP (99.24%) and CENTRIST (99.24%) method performed well because of their holistic nature and ability of detecting microstructures. NABP descriptor performed well (99.08%) because of it's ability to noise adaptiveness and discriminative nature as mentioned before. Again, among ternary methods DTCTH (99.2%) and LTP (99.16%) performed better than LAID (97.26%) as LAID may be affected by out-layer for it's median value based threshold.

Despite all these efforts it can be noted recognition accuracies for Bangla handwritten isolate characters are not up to the mark. For building a complete working HCR, we need to have a recognizer that recognize isolated characters excellently as for HCR other tasks such as segmentation of texts, sentences and words will make the task for isolated character classifier hard. Therefore, for obtaining better results more advanced and effective learning techniques like deep learning is needed to be used.

Chapter 4

Deep Convolutional Neural Network (D-CNN)

4.1 Introduction

Deep learning is a form of machine learning that is inspired by the processing of human brain although the processes of deep learning is vaguely related with neuroscience. Different types of deep learning networks are used for different takes. For example, for image data, Convolutional Neural Network is used, Recurrent Neural Network is for temporal data etc. Deep learning has gained popularity because of its increased performances on various structured and un-structured data as compared to the typical machine learning algorithms. The reason of such performance gain is that we have a large number of data available now-a-days. Deep learning enables us to get accuracy gain with the increased number of data whereas typical state-of-arts machine learning algorithms does not increase the accuracy with increased data. However, for hard problems like handwritten character recognition Deep Convolutional Neural Network (DCNN) can be a handy tool as it is shown to give better performance in pattern recognition and image classification. A DCNN model consists of some layers. The three main layers used in DCNN are input, convolutional, pooling, fully connected and output layer. Now-a-days more and more deeper layers are becoming available because of increased computational capabilities. The layers in DCNN works in a way that the shallower layers are responsible for finding the major characters of an image like colors and shades, the deeper networks find out the microstructures form an image. The layers of DCNN along with some other important terms are described briefly in the following sections.

4.2 Layers of DCNN

4.2.1 Input Layer

In this layer, the image is feed to the DCNN network as raw pixels. Images can be RGB or grayscale. The pixel values are typically normalized to be in range of 0 to 1 in place of 0 to 255. This makes learning faster and helps to get better performance.

4.2.2 Convolutional Layer

Convolution operation is the building block of DCNN. An example of convolution operation is given in Fig. 4.1. As can be seen from the figure, convolution operation takes the element wise multiplication of the left top 3 x 3 pixels with the 3 x 3 filter and add these multiplied values which is then put as the first value of the output matrix. Then, this same operation is performed by shifting the 3 x 3 cell of the input image to right. The number of cells to be shifted is called a 'Stride' and typical value of stride is one for convolutional layer. Fig. 4.2 gives an example of convolution operation for a RGB image. For

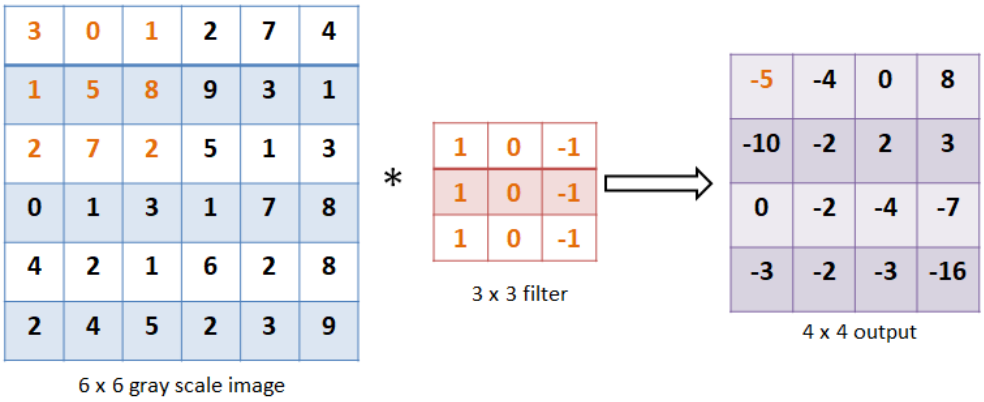


Figure 4.1: Convolution operation

RGB image the shape of the input is (height x width x 3) where the value 3 indicates the number of channels. The filter shape in this case is taken as 3 x 3 x 3. Each slice of the filter is convoluted with the corresponding slice of the input image after that all these 3 values are added and considered as output. More than one filter can be used in either cases. While using more than one filters the output matrixes are stacked together. The output shape for both these cases are given as:

$$(n - f + 1) \times (n - f + 1) \times n_c$$

Here, n and f is the height and width of input image and filter correspondingly. Nc denotes the number of filters used.

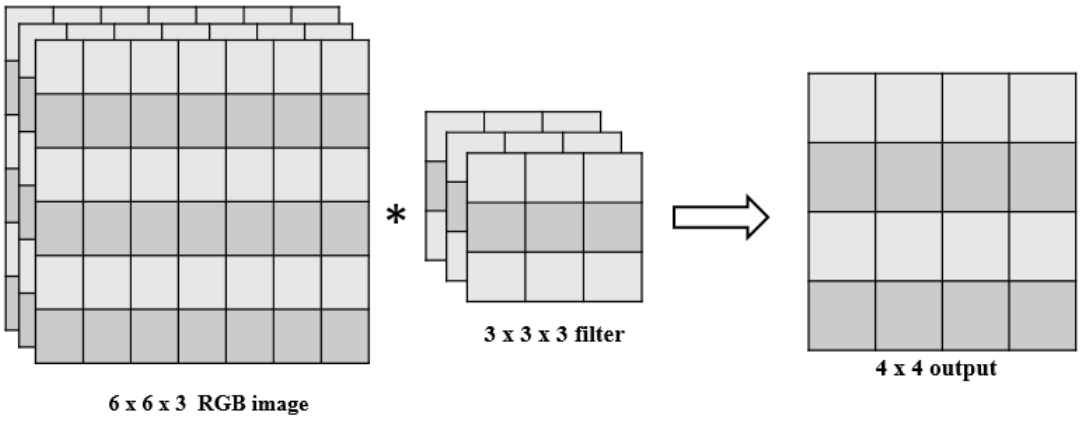


Figure 4.2: Convolution operation for RGB image

We can see that; after convolution operation the image shrank a little bit. If a very deep network is used this can be problematic also putting less emphasis on the border pixels may lead to waste of valuable information. For avoiding such problems padding is used to preserve the input shape. Based on preserving of the shape, convolution operation can be divided into two types. Those are:

- Valid convolution: shape is not preserved i.e., padding is not used.
- Same convolution: shape is preserved using padding

Fig. 4.3 gives an example of same convolution with one layer of zero padding. For convolution with padding, the output shape is given by:

$$(n + 2p - f + 1) \times (n + 2p - f + 1) \times n_c$$

here, n and f are the height and width of input image and filter respectively. Nc is the number of filters used and p is the number of layers of padding. A point can be noted that, by changing the value of p we can yield desired shape of output. The filters used in DCNN are basically initialized randomly and considered as learning parameters. In the convolution layer two other task is done to make it possible for a DCNN to work. After the convolution operation a bias value is added to the output matrix to give it a little offset. The resultant value is than passed through an activation function to add non-linearity. Without this activation function the model would act same no matter we add how many layers. Some activation function used for this purpose are:

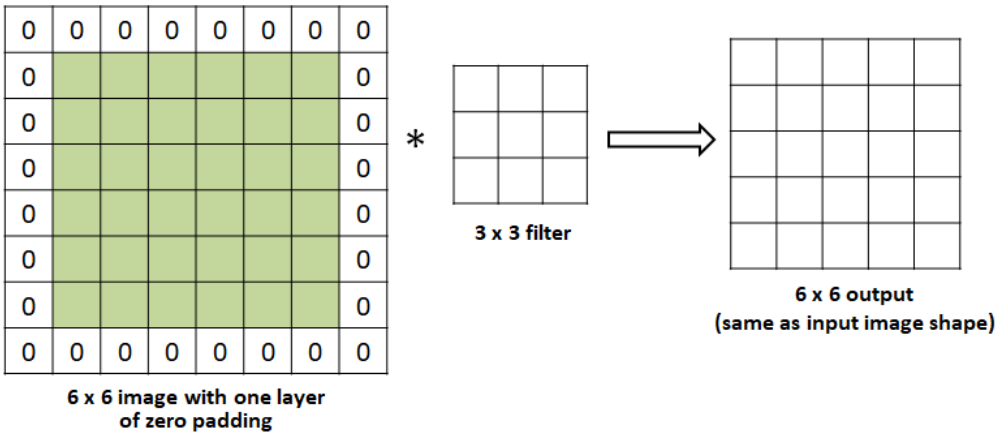


Figure 4.3: Zero padding to prevent input size

Sigmoid function:

Sigmoid function is defined by the following equation:

$$A = \frac{1}{(1 + e^{-z})} \tag{4.1}$$

Here, e is a constant and e = 2.71828. z is the value obtained after adding the bias value with the convolution output. This function gives output in the range of 0 to 1.

Tanh function:

Tanh function is defined by the following equation:

$$A = \frac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})} \tag{4.2}$$

Here, e and z is same as previously defined. This function gives output in the range of -1 to 1. This function works better than sigmoid function is most cases. Sigmoid function is typically used in binary classification.

ReLU activation function:

ReLU or Rectified Linear Unit is defined by

$$A = max(0, z) \tag{4.3}$$

here, a and z bears the conventional meaning. This activation function converts the input into a positive value. This is mostly used activation function and also used for this study.

Leaky ReLU function:

This function is similar to ReLU except that instead of the minimum value being 0, it is chosen to be a small number. This function is defined as:

$$A = max(k, z) \tag{4.4}$$

Here, a and z is same as previous and k is a constant close to zero.

4.2.3 Pooling Layer

This layer takes output of the previous layer and down samples this by choosing prominent features. This layer includes two hyper-parameters and no learning parameters. The hyper-parameters are shape of filter and stride. Most common shape of filter is 2 x 2 with a stride of 2. By reducing the shapes, pooling layer makes the whole process of DCNN computationally efficient. Using a pooling layer generally decreases the height and width, but, the number of filters remains the same. Pooling layers generally does not apply padding. The output shape after a pooling operation is given by:

$$\frac{(n_h - f)}{s} + 1 \times \frac{(n_w - f)}{s} + 1 \times n_c$$

Here, nh , nw, nc denotes height, width and number of filter of the input. 's' is the number of stride and f is shape of filter. Example of two mostly used pooling operation is given below:

MAX pooling:

Max pooling takes the maximum value from a given filter size. As we can see from Fig. 4.4, 2 x 2 max pool filter is used with stride of two. Starting from the top-left hand 2 x 2 section the maximum value of this section is calculated and placed in the output. The filter is than moved 2 cells and from right top 2 x 2 pixel values maximum is calculated. The same operation is done for the bottom left and right chunks of this image. Max pooling basically keeps the most important or largest features. It is popularly used pooling operation now-a-days.

AVERAGE pooling:

Average pooling works the same way max pooling works except that from a given shape of filter the average value is taken instead of the

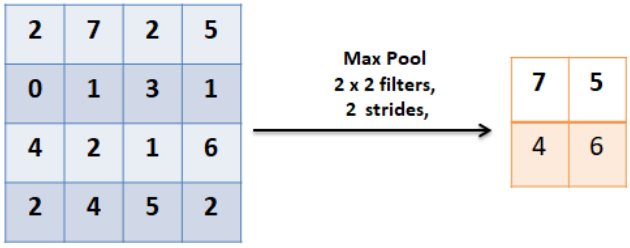


Figure 4.4: MAX pooling operation

maximum value of max pooling. Average pooling considers the magnitude of all the features of a given range. Average pooling is not that much used. Fig. 4.5 gives an example of average pooling.

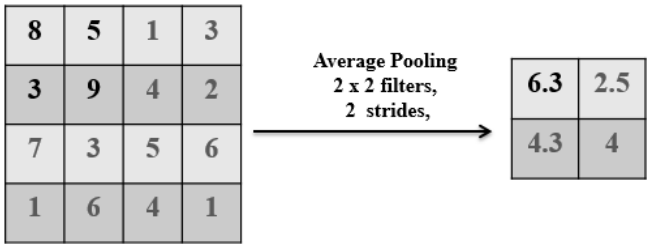


Figure 4.5: Average pooling operation

4.2.4 Fully Connected Layer

After a set of convolution and pooling operation is performed on an input image, the output is then flattened to have a one dimensional shape instead of two dimensional matrices, which is then feed to the fully connected layer. fully connected layer is like hidden layer of a neural network that consists of a number of nodes. The output we get after passing the flattened data through one or more fully connected layer is feel to the softmax output unit.

4.2.5 Output Layer

A softmax output unit calculates probability of a given image to be in a certain class. This unit assign a probability value to all the possible classes which adds up to 1. The class containing the largest value is the predicted class. One-hot representation takes these probability values and keeps the largest value as 1 and rest of the values as zero.

4.3 How DCNN Works

DCNN works pretty much the same way a neural network work. In the case of DCNN models the filter values and bias values of each layers acts as learning parameter. Fig. 4.6 depicts forward propagation of one layer of DCNN. Using forward propagation for each layer in the same depicted, predicted output is evaluated. After that, using this predicted label and true label, loss function is calculated and our objective is to optimize the learning parameters such that this loss function is minimized. Using some optimization function, the backward propagation step modifies the learning parameters so that the loss function is minimized. Such one step of forward and backward propagation is called an epoch. A number of epochs made the model to fit given data. This is how a DCNN works.

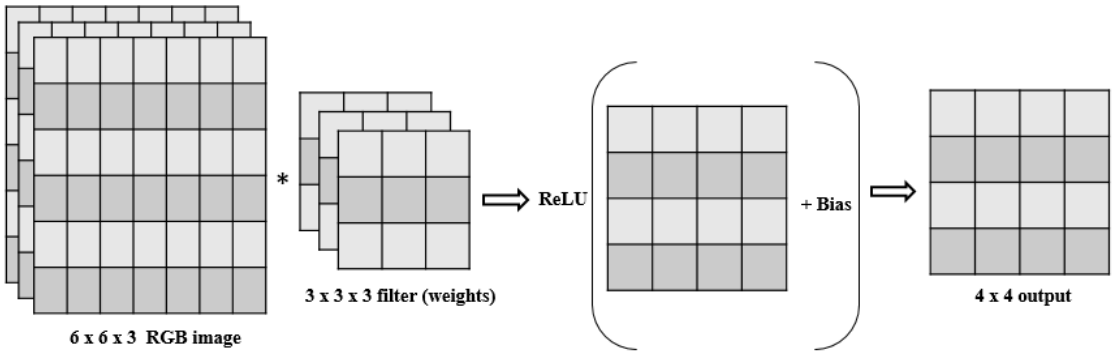


Figure 4.6: A layer of DCNN

4.4 Optimization Algorithms

The optimizing learning parameters, some optimization algorithms are used. More or less every optimizer work by using gradient values. Learning rate of optimization is an important hyper parameter that ensures fast convergence and avoiding divergence. Some mostly used optimization algorithms are described below:

4.4.1 Gradient Descent

This algorithm takes the derivative of the weights and biases and reduce the derivative value from present weights and biases. This approach of optimization is analogous to climb down a hill. If we are on the top of a hill i.e., the loss function is high, we need to take steps

towards the way that will take us to the bottom of the hill i.e., minimized cost. The slope or derivative of the loss function at a certain point defines where to go for going downhill and the learning rate defines the size of our steps. For optimizing weights, the derivative of loss function is calculated with respect to weights and with respect to bias for optimizing bias values. The overall algorithm can be given by:

Repeat until convergence:
{
Weights = Weights – learning rate x derivative of loss function with
respect to weights
Bias = Bias – learning rate x derivative of loss function with respect to biases
}

4.4.2 Gradient Descent with Momentum

In the previous case it can be noted that, gradient or derivative value with learning rate is used. Using only the gradient value may cause more oscillation and hence, take more time to fully converge to the minima. So, in gradient descent with momentum, exponentially weighted average of the gradients is used. Exponentially weighted average takes into account the past values and thus makes the propagation through the minima smoother. This process can be written as:

$V_{dw} = 0$
$V_{db} = 0$
Repeat until convergence:
{
$V_{dw} = \beta * V_{dw} + (1 - \beta) * dw$
$V_{db} = \beta * V_{db} + (1 - \beta) * db$
$Weights = Weights–learningrate \times V_{dw}$
$Bias = Bias–learningrate \times V_{db}$
}
Here, beta is a hyper parameter and generally beta=0.9 that takes into account past 10 values.

4.4.3 Root Mean Square Prop (RMSprop)

This is another variation of gradient based optimization. The algorithm of RMSprop can be given as:
$Sdw = 0$
$Sdb = 0$
Repeat until convergence:
{
$V_{dw} = \beta \times V_{dw} + (1 - \beta) \times dw$
$S_{dw} = \beta_2 \times Sdw + (1 - \beta_2) \times dw2$
$S_{db} = \beta_2 \times Sdb + (1 - \beta_2) \times db2$
$Weights = Weights - \frac{(learningrate \times dw)}{root(Sdw+e)}$
$Bias = Bias - \frac{(learningrate \times db)}{\sqrt{(Sdb+e)}}$
}

Here, beta1 is a hyper parameter, e is a constant added to avoid division by zero. As RMSprop takes the square root of the weights and biases, it makes the large value larger and small values relatively less large. This ensures that weights or biases having large slope will be divided by a relatively large number and thus reduce the peak slopes and make it smooth. On the other hand, relatively small slopes get divided by a relatively small value hence making the net effect less. Thus RMSprop smooth out the oscillation and enables fast training.

4.4.4 Adam Optimizer

Adam stands for Adaptive Moment Estimation. Adam is the most popular optimizer which has deployed in many sectors and shown to provide better results. It combines both RMSprop and Gradient descent with momentum and gives better result.

$V_{dw} = 0$
$V_{db} = 0$
$S_{dw} = 0$
$S_{db} = 0$
Repeat until convergence:
{
$V_{dw} = \beta_1 * V_{dw} + (1 - \beta_1) * dw$
$S_{dw} = \beta_2 * S_{dw} + (1 - \beta_2) * dw^2$
$V_{db} = \beta_1 * V_{db} + (1 - \beta_1) * db$

$$S_{db} = \beta_2 * S_{db} + (1 - \beta_2) * db^2$$
$$Weights = Weights - \frac{(learningrate \times V_{dw})}{\sqrt{(S_{dw} + e)}}$$
$$Bias = Bias - \frac{(learningrate \times V_{db})}{\sqrt{(S_{db} + e)}}$$

} Here, all the variables and constant used are same as previously defined. For adam optimizer bias correction is used. Values of Vdw, Sdb, Vdw and Sdb are corrected before using.

4.5 Problem of Overfitting

Overfitting is one of the major problems a DCNN model can have. Overfitting a model implies that the model is fitting the training data too well. That leads to a model that is not generalized properly. Such model gives a very good performance in training set but performs relatively less on test data. For avoiding this problem some measures can be taken. Such as decreasing the complexity of the model, use more data to train the model, decrease the number of features, use regularization technique that can decrease the net effect of weights etc. For this study an effective regularization technique, namely, dropout regularization is used. The process of regularization is described briefly in the next section.

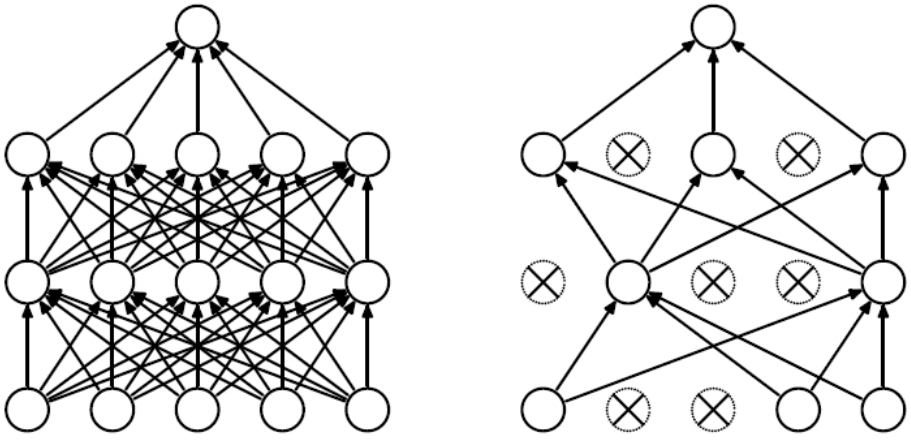


Figure 4.7: Dropout regularization (left: neural network without dropout, right: with dropout)

4.5.1 Dropout Regularization

Fig. 4.7 depicts the technique of dropout regularization in neural network. To implement dropout each layer of a DNN or DCNN is

given a probability value called 'keep probability' or 'keep prob'. Keep prob of 0.7 means that each node in each layer have 30% chance to be deactivated while training, other 70% will be kept as it is. This technique brings randomness in the model and hence decrease the chance of overfitting.

4.6 BanglaNet-14

The proposed model is named BanglaNet-14 being inspired by the essence of our language and considering the number of layers. BanglaNet-14 has been used for evaluating the performance for all categories of Bangla characters also a result is evaluated by combining all types of characters together. For experimenting hyper parameters settings was same for all evaluations. The next Section provides a detailed description of the proposed model.

4.6.1 Architecture of BanglaNet-14:

Figure- 20 gives an overall detailed structure of BanglaNet-14. Table 4.1 gives the parameter settings for each layers. Each steps are described below:

Input layer:

for this layer all the images of dataset are resized to have similar shapes. There are variations of shapes in different images of each dataset (except CAMTERdb 3.1.1). however, each RGB images are resized to have 112 x 112 x 3 shapes. Then, all the pixel values are normalized dividing by zero. This is done to ensure a uniform distribution of input data and faster convergence. As seen from Fig. 4.8, the input shape is 112 x 112 x 3.

Convolution Layer 1:

this layer takes the 112 x 112 x 3 input images and convolve it with 32 filters of 3 x 3 x 3 shape (as the number of channels must match). As same convolution is applied, the output shape remains the same. ReLU activation is then applied to this result.

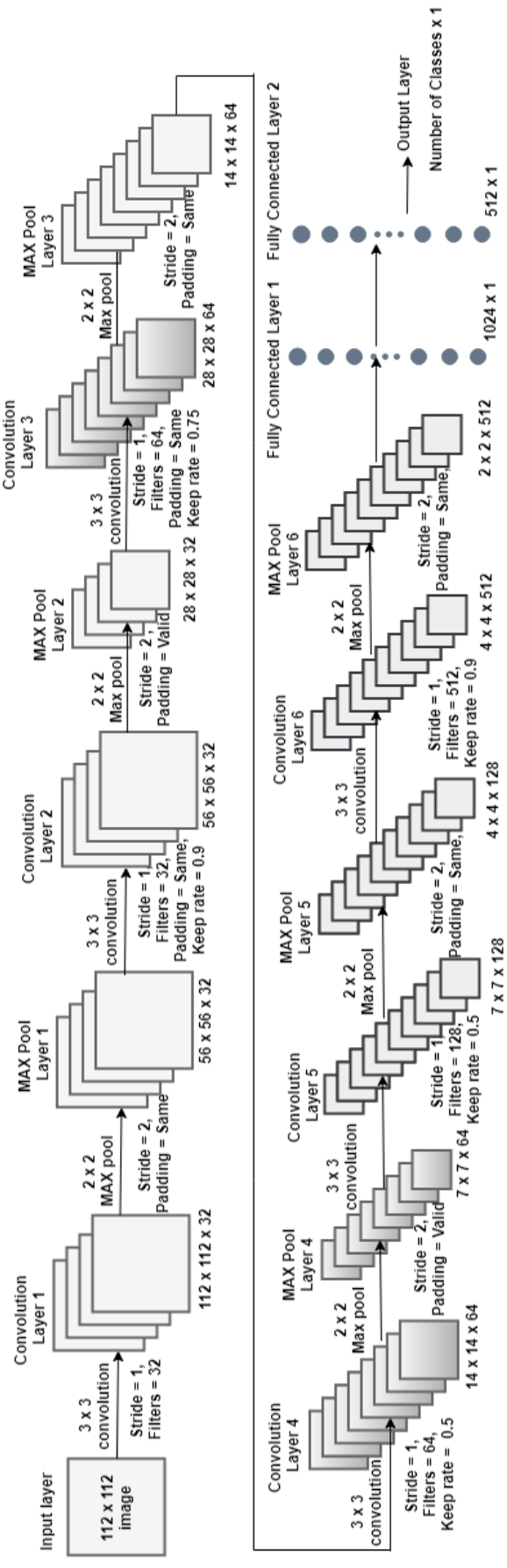


Figure 4.8: Complete architecture of the proposed DCNN model (BanglaNet-14)

Max Pool Layer 1:

this layer takes output from the previous layer and apply max pool to it. Same padding is used for some max pool operation to maintain the shape. For max pooling 2 x 2 filter with stride of 2 is used.

Convolution Layer 2:

this layer takes the output from previous max pool layer and convolve it with 32 filters of 3 x 3 x 32 shape with stride of 1. In this layer dropout with 90% keep rate is used. After applying ReLU activation output of this layer is 32 x 32 x 32.

Max pool layer 2:

this layer takes output from previous layer and down samples it with a max pool operation of 2 x 2 filter and stride of 2. Valid pooling is used in this case that halves the height and width.

Convolution layer 3:

64 filters of 3 x 3 x 32 shape is used in this layer. After convolution, ReLU activation is applied. In this layer dropout of 75% keep rate is applied.

Max pool layer 3:

this layer applied a 2 x 2 pooling with stride of 2. Same pooling is applied in this layer. Output height and width remains the same.

Convolution layer 4:

this layer applied 64 filters of 3 x 3 x 64 shape. Same convolution is applied with a dropout of 50% keep rate. ReLU activation is applied at last.

Max pool layer 4:

2 x 2 max pool with stride of 2 is used in this case. Valid pooling is used. As a result, the output shape is halved to 8 x 8 keeping the number of channels same.

Table 4.1: Summary of each Layer of BBCNet-15

Layer	Output Shape	No. of parameters
Convolution 1	(None, 112, 112, 32)	896
Max Pool 1	(None, 56, 56, 32)	0
Convolution 2	(None, 56, 56, 32)	9248
Max Pool 2	(None, 28, 28, 32)	0
Convolution 3	(None, 28, 28, 64)	18496
Max Pool 3	(None, 14, 14, 64)	0
Convolution 4	(None, 14, 14, 64)	36928
Max Pool 4	(None, 7, 7, 128)	0
Convolution 5	(None, 7, 7, 128)	73856
Max Pool 5	(None, 4, 4, 128)	0
Convolution 6	(None, 4, 4, 512)	590336
Max Pool 6	(None, 2, 2, 512)	0
Fully Connected 1	(None, 1024)	2098176
Fully Connected 2	(None, 512)	524800
Output	(None, no. of classes)	512 x no. of classes

Convolution layer 5:

128 filters of 3 x 3 x 64 is applied in this layer with stride of 1. As activation function ReLU is used. Same convolution is used, so the output shape is 7 x 7 x 128. Dropout of keep rate 50% was added to regularize the model.

Max pool layer 5:

same pooling with 2 x 2 shape and stride of 2 is used. Output shape is 4 x 4 x 128 of this layer.

Convolution layer 6:

512 filters of shape 3 x 3 x 128 is used with stride 1 with 90% keep rate. ReLU activation is used in this layer.

Max pool layer 6:

2 x 2 filter with a stride of 2 is used. In this layer same pooling is used.

Fully connected Layer 1:

these last two layers are like neural networks. Results from the previous layer is flattened and feed to each 1024 nodes of this layer.

Flattened output from previous layer is multiplied with the weights assigned to each nodes. After that, a reLU activation is applied and a bias value is added. Dropout with 90% keep rate is applied in this layer.

Fully connected Layer 2:

output from the previous fully connected layer is the input of this layer. Output from previous layer acts as activation function and are multiplied with the weights of this layer. Softmax activation unit is applied in this layer. Output of this layer is probability values assigned to be one of the classes of input data.

Fig. 4.9 gives the output of each layers of proposed improved DCNN model. Output of only 4 filters are shown here. Each layers have such 32, 64 or 512 filters detecting various features. As can be seen from the image, first convolution layer basically detected outwards edge like features and activated the areas containing edge like nature. The following max pooling layer basically magnified the those areas. Similarly, the following layers have detected inward edge like features from the image. More advanced layers have detected micro and macro structures form the image until a feature map is formed for a certain type of images. Thus every convolutional layer detected certain type of features using quite a lot number of filters. Using more shallower layers like used in LeNet-5, will yield lesser number of filters and learning parameters and as result the performance will suffer. Using more deeper DCNN model will be problematic due to the issue of vanishing or exploding gradient. With more deeper layers, vanishing gradient causes the gradient of final layers to be too small to effectively turn off the learning of the model. Exploding gradient causes the gradient to be too large that will cause the loss function to increase or make the gradient NaN. As relu function is used as activation unit, vanishing gradient should not be a problem for the proposed model. However, exploding gradient can occur is very deeper network are used. For these reasons, the proposed model is just right for the classification problem of concern. It is tuned with just right amount of hyper-parameter settings and regularization. If compared with LeNet-5, it contains far less number of features and learning parameters which will detect much fewer and only some basic features which is not sufficient for such a complex classification task. So using BanglaNet-14 gives advantage over simpler models or more complex model in terms of

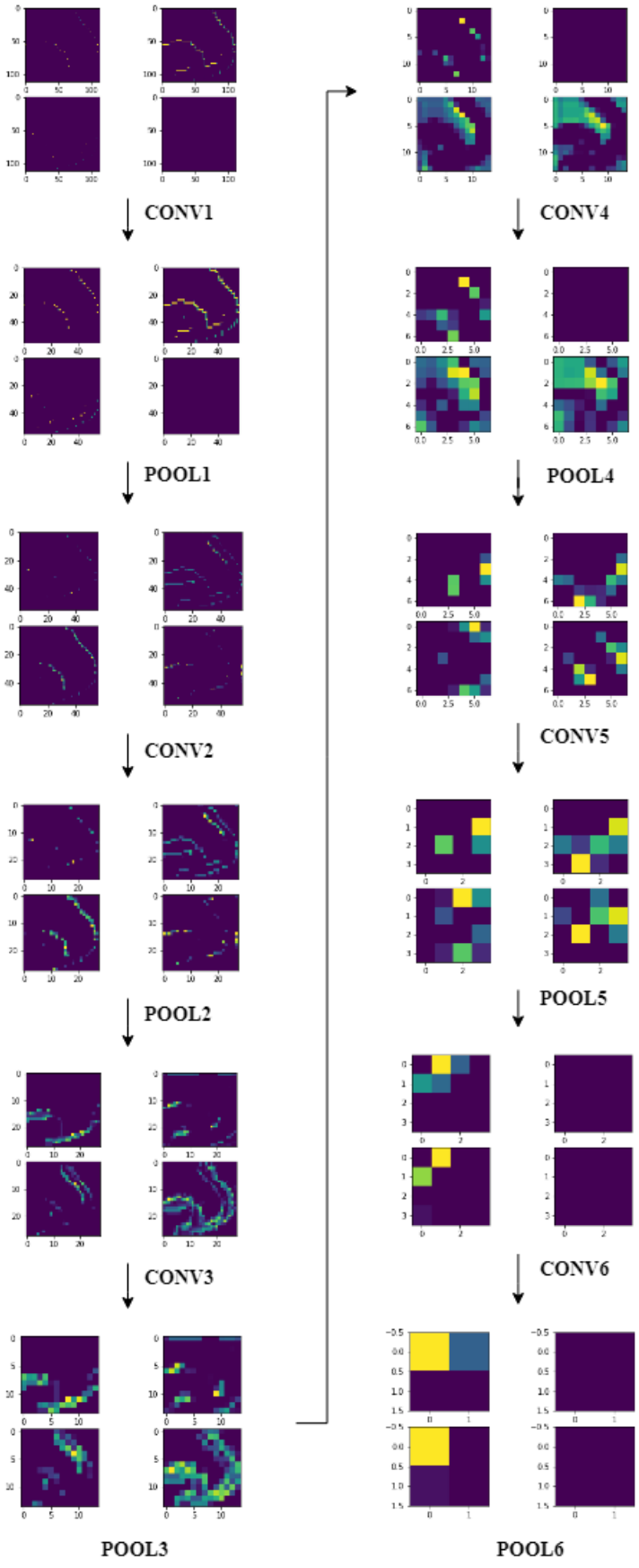


Figure 4.9: Output of each layers for a sample character

just right learning parameters, right amount of features and avoiding exploding or vanishing gradient problem.

4.7 Summary

In this Chapter, pros anand ad cons of DCNN is discussed along with the architecture of proposed DCNN. A series of convolution, activation and pooling layers is mainly used to extract features from the given image data. Using deeper model can increase performance of a given classification task with the cost of increased computation.

Chapter 5

Experimental Evaluation

5.1 Introduction

Using the proposed model (4.6) three different benchmark data set (as described in 5.1) for Bangla basic characters, compound characters and digits are evaluated and proven to give better results than existing methods. In this Chapter, experimental results obtained for these three dataset is discussed along with used implementation environment.

5.2 Dataset

For the purpose of evaluating the proposed DCNN model, benchmark datasets for Bangla basic characters, compound characters and digits were used. This chapter describes the used dataset and use of the data for evaluation briefly for better understanding. Also, usage of these dataset for experimental evaluation is shown in Table 5.1. For evaluation purpose, CMATERdb dataset is used. CMATERdb is basically a pattern recognition dataset repository prepared at the Center for Microprocessor Applications for Training Education and Research (CMATER) which is a research laboratory of Jadavpur University located at Kolkata in India. This database contains three different datasets for Bangla basic characters, compound characters and digits which are freely available for academic use. The image data of these datasets have variations in style and some images are little bit noisy. Also the images of characters are not co-centric and size of images varies (only the dataset for digits provides all same shaped images). Table 5.1 provides the properties of each dataset along with the usage of the data for evaluating proposed system. The following sub-sections

Table 5.1: Usage of Three Datasets for Experimental Evaluation

Dataset name	**CMATERdb 3.1.2 (Basic characters)**	**CMATERdb 3.1.3.1 (Compound characters)**	**CMATERdb 3.1.1 (Digits)**
Total number of classes	50	171	10
Total training images	12000	33404	4000
Training images per class	240	150 to 290	400
Total test images	3000	8389	2000
Test per class	60	40 to 60	200

provides an overview of all three datasets.

5.3 Bangla Handwritten Basic Character's Dataset

5.3.1 CMATERdb 3.1.2:

There are total 50 basic characters in Bangla language including 39 consonants and 11 vowels. For the purpose of testing the proposed model with Bangla basic characters, CMATERdb 3.1.2 dataset was used [4, 5]. This dataset contains two folders, one for the train images, one for the test images. Each folder contains 50 sub-folders. Each of these 50 sub-folder contains images for the 50 classes and the label of each class is the name of these sub-folders. In 'train' folder each sub-folder contains 260 images making a total of 12000 images. On test folder each sub-folders contains 60 images making a total of 3000 images. The train images are used to train the model and test images are used to validate the model. Later on, the test images were used to test the model and evaluate precision, recall and F1 score for each classes. Fig. 5.1 gives some sample images from the dataset.

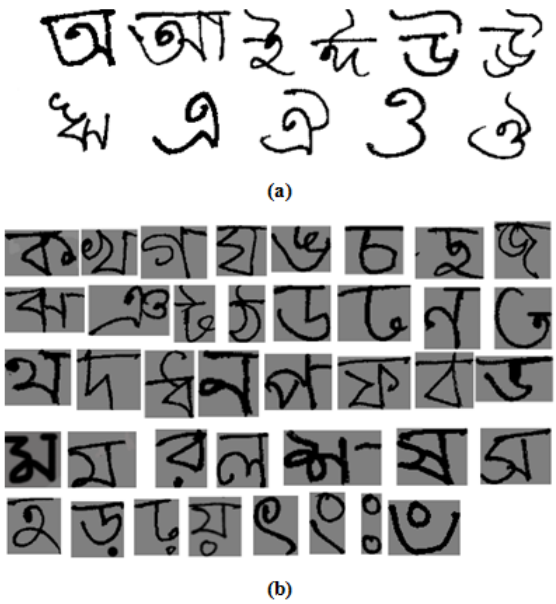


Figure 5.1: Sample images from CMATERdb 3.1.2 dataset (a) vowels (b) consonants

5.4 Bangla Handwritten Compound Character's Dataset

5.4.1 CAMTERdb 3.1.3.1:

There are around 300 compound characters available in Bangla language. Not all of these compound characters are used very frequently. Based on the usage of compound characters, CMATERdb 3.1.3.1 dataset contains total 171 compound character classes [43]. This is most probably the only dataset that contains that many classes. Fig. 5.2 shows some example images from the dataset. However, the arrangement

Figure 5.2: Sample images from CMATERdb 3.1.3.1 dataset

of training and test images are same as the basic character dataset. Only, in this dataset each sub-folder in training folder contains 150 to 290 training images and in test folder, each classes contains 40 to 60 images. Figure 10 gives some example characters from the dataset. This dataset is very challenging not only for the number of classes, but also for the fact that there are some compound characters that have more than one representations. To be exact, 27 classes among all 171 classes have more than one representation for the same character. Fig. 5.3 shows some example of some of such characters. For evaluation purpose the protocol applied for training, validating, testing and measuring recall, precision and F1 score is same as basic character's.

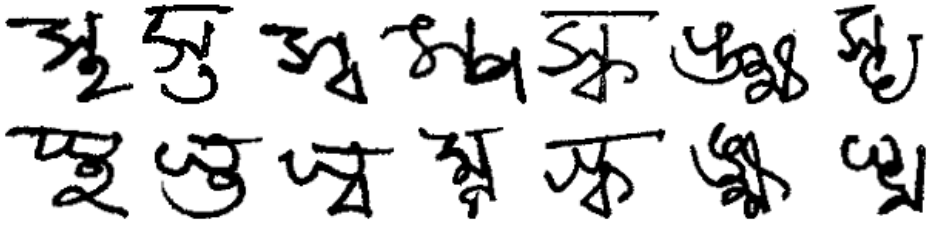


Figure 5.3: Compound characters with multiple representation (same column represents same characters)

5.5 Bangla Handwritten Digit's Dataset

5.5.1 CMATERdb 3.1.1

There are total 10 Bangla digit classes. For evaluating the performance of digits CMATERdb 3.1.1 dataset is used [38, 39]. This dataset contains total 6000 images. Fig. 5.4 shows some example images from the dataset. For evaluation purpose, 4000 images were used as training image and 2000 images were used as test images. All the 4000 and 2000 images were divided into 10 sub-folders of 10 classes. The model was trained using the train folder. Test folder was used to validate the model as well as calculating the F1 score by the confusion matrix.

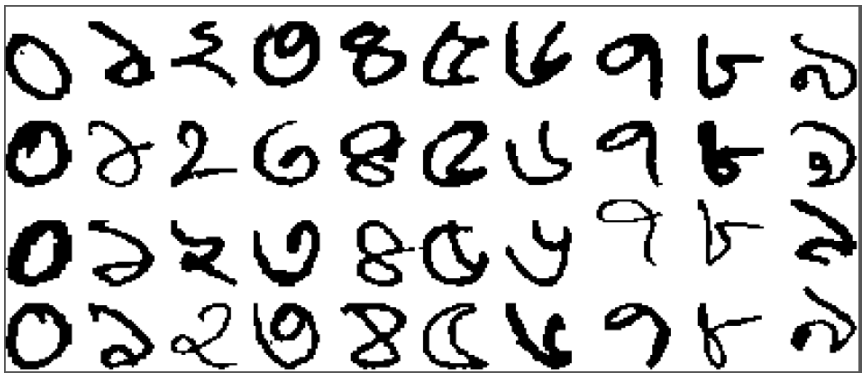


Figure 5.4: Sample images from CMATERdb 3.1.1 dataset

5.6 Dataset for Bangla basic characters, compound characters and digits

For a complete HCR a classifier that can detect all kind of Bangla characters including basic characters, compound characters and digits is needed. For this purpose, 3 different datatsets are used to evaluate the proposed improved DCNN model. Those datasets are described

in the following subsections. Table 5.2 provides the usage of these datasets.

5.6.1 CMATERdb

This dataset is formed by combining CMATERdb 3.1.2, CMATERdb 3.1.3.1 and CMATERdb 3.1.1 datasets. Total number of classes are 231 of this combined dataset. Table 5.2 shows the settings for training and test images. Variation of the number of samples per classes makes this dataset quite challenging and prone to overfitting. However, this dataset contains the most number of classes of compound characters.

5.6.2 BanglaLekha-Isolated

This dataset consists of 84 classes of Bangla isolated characters' sample [22]. Among these 84 classes 50 are basic characters, 10 are digits and rest of the 24 classes are frequently used compound characters. A total of 166105 data samples are included in this dataset. Unlike CMATERdb dataset, this dataset is not divided into training and test folders. Each folders contains about 2000 image samples which were divided into train and test samples at training time. As training data 80% of the images and as test data 20% of the images are used. Fig. 5.5 shows some example images from the dataset.

Figure 5.5: Sample characters and digits from BanglaLekha-isolated dataset

Table 5.2: Usage of Datasets for Experimental Evaluation

Dataset name	**CMATERdb**	**BanglaLekha-isolated**	**Ekush**
Total number of classes	231	84	122
Total training images	49404	132884	31196
Training images per class	150 to 400	1580 to 1590	2540 to 2560
Total test images	13389	33221	55053
Test per class	40 to 200	390 to 400	440 to 460

5.6.3 Ekush

Ekush dataset contains a total of 122 classes among which 50 basic characters, 10 digits and 62 frequently used compound characters are included [25]. Total number of sample is 367018. Whole dataset is divided into two parts that contains equal numbers of written characters from male and female participates. However, for using this in our proposed model, both parts are combined. Each classes contains about 3000 images. As training data, 85% of all images are used and as test data 15% data are used. Fig. 5.6 gives some examples from the dataset.

Figure 5.6: Sample characters, modifires and digits from Ekush dataset

All the aforementioned datasets were used to evaluate the performance of Bangla basic characters, compound characters and digits. However, for the shake of building a complete Bangla HCR we need to evaluate the proposed model on all the previously described classes all together. For this purpose, CMATERdb 3.1.2, CMATERdb 3.1.3.1 and CMATERdb 3.1.1 datasets were combined to form a dataset to calculate the accuracy and F1 score on all these 231 classes. The experimental details with the described settings are given in following Sections.

5.7 Implementation details

In the case of DCNN models the filter values of each layers acts as learning parameter. Using forward propagation, predicted output is evaluated. After that, using this predicted label and true label, loss

function is calculated and our objective is to optimize the learning parameters such that this loss function is minimized. Using some optimization function, the backward propagation step modifies the learning parameters so that the loss function is minimized. Such one step of forward and backward propagation is called an epoch. A number of epochs made the model to fit given data. For BBCNet-15 categorical cross entropy function is used as loss function and to optimize it adam (Adaptive Moment Estimation) optimizer [44] is used (see 4.4.4). Categorical cross entropy is defined using (5.1).

$$H_{\hat{y}} = -\sum_{i} \hat{y} log(y_i) \tag{5.1}$$

For implementing the proposed model, the following environmental settings are used:

- Operating system: Windows 7 professional
- CPU: Intel core i5-3470 with 8 GB RAM
- GPU: NVIDIA GeForce 1050 Ti 4GB
- Environment: Keras framework [45] with Tensorflow [46] as backend
- Programming language: python
- IDE: Spyder

5.8 Time and Space Analysis

For any DCNN model, time and space analysis is of vital importance as time and space requirement is much higher for it than state of arts machine learning techniques. In this Section time and space required for the implementation of BanglaNet-14 is given.

The main time required for this model is while training. After training the trained model is saved and used for test new data. Table 5.3 gives the time to train different models and Table 5.4 gives data regarding testing a number of images. As can be noted dataset containing less than 12000 images trains within 30 minutes which is not that much. However, training LeNet-5 model takes less time than training BanglaNet-14 since the later has considerably more parameters than the first. Also the performance gain is visible for BanglaNet-14. The

datasets containing 50k or 60k samples takes below 2 hours. However, dataset containing 1M to 3M images takes a considerable amount of time to finish training. After the model is trained, it is ready to take test images. Testing images requires for each images is 0.6 milliseconds to 1 milliseconds regardless the DCNN model or test data used here.

Table 5.3: Training time (m = minutes, s = seconds)

Dataset	**Training time (BanglaNet-15)**	**Training time (LeNet-5)**
CMATERdb 3.1.2	27 m 51 s	13 m 53 s
CMATERdb 3.1.3.1	79 m 19 s	45 m 25 s
CMATERdb 3.1.1	8 m 18 s	6 m 3 s
CMATERdb	117 m 40 s	75 m
BanglaLekha Isolated	340 m 17 s	282 m 18 s
Ekush	809 m 14 s	684 m 54 s

Table 5.4: Testing time (m = minutes, s = seconds)

Dataset	**Test time: BanglaNet-15 (fetch + test), test**	**Test time: LeNet-5 (fetch + test), test**	**No. of images**
CMATERdb 3.1.2	4.61 s, 4.19 s	2.33, 1.91	3000
CMATERdb 3.1.3.1	5.73 s, 4.63 s	6.29, 5.24	8389
CMATERdb 3.1.1	1.737 s, 1.43 s	1.77, 1.46	2000
CMATERdb	20.06 s, 18.41 s	10.26, 8.61	13389
BanglaLekha Isolated	122.32 s, 103.34 s	131.01 s, 112.61 s	166105
Ekush	255.04 s, 211.12 s	250.81 s, 205.42 s	362732

Table 5.5 shows the space required for each images for BanglNet-14 and LeNet-5. Space required to process each images using BanglaNet-14 is 6.1 MB while training and 3.05 MB while testing since testing requires only one pass. Whereas training and testing requires three times lower memory. It is because of the lower number of learning parameters. However, 3 or 6MB per images is not a lot of memory, so, the trained model can be easily used in smartphones. Model with more deeper layers could have increased this required space.

Table 5.5: Space usage per image

Layer	**Memory per image: BanglaNet-14**	**Memory per image: LeNet-15**
Input	112*112*32 = 37K	112*112*32 = 37K
Convolution 1	112*112*32 = 401K	108*108*6 = 69k
Pool 1	56*56*32 = 100K	54*54*6 = 17k
Convolution 2	56*56*32 = 100K	50*50*16 = 40k
Pool 2	28*28*32 = 25K	25*25*16 = 10k
Convolution 3	28*28*64 = 50K	-
Pool 3	14*14*64 = 12K	-
Convolution 4	14*14*64 = 12K	-
Pool 4	7*7*128 = 6K	-
Convolution 5	7*7*128 = 6K	-
Pool 5	4*4*128 = 2K	-
Convolution 6	4*4*512 = 8K	-
Pool 6	2*2*512 = 2K	-
Fully Connected 1	1K	0.12k
Fully Connected 2	0.5K	0.084k
Total Usage:	762.5K * 4 bytes = 3.05 MB 2 * 3.05 = 6.1 MB (for both passes)	173.2K * 4 bytes = 0.7 MB 2 * 0.7 = 1.4 MB (for both passes)

5.9 Result and Discussion: CMATERdb 3.1.2

The model is trained for 100 epochs. Different learning rates ranging from 0.0009 to 0.0001 was considered for training. However, it is seen that using learning rate 0.0005 trained the model relatively better than other learning rates. So, the detailed evaluation of the model is shown using learning rate 0.0005. Fig. 5.7 (pg. 61) gives the learning curve of the model with respect to loss function. It can be noticed, Lowest loss value obtained for training data is 0.0234 and for test data is 0.1032. Fig. 5.8 (pg. 61) gives the test and training accuracy of the model with respect to number of epochs used. As can be seen, curve for training and testing is almost overlapping which indicates a generalized model. Maximum training accuracy obtained was 99.24% and test or validation accuracy is 97.67%. Fig. 5.10 (pg. 62) depicts a confusion matrix for all 50 classes. Precision, recall and F1 score is computed for each classes using the values gained from the confusion matrix which is shown in Fig. 5.9 (pg. 62).

Classes are labeled from 1 to 50 while implementing. However, for better illustration printed characters of each classes are given in Fig. 5.9. Average precision obtained is 0.98, average recall is 0.98 and

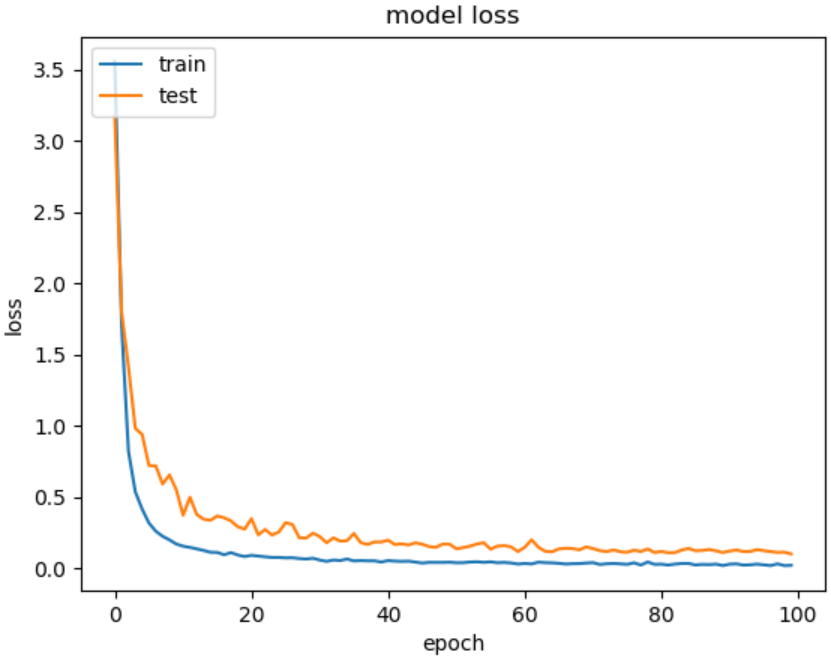


Figure 5.7: Train and test loss with respect to no.of epochs for CMATERdb 3.1.2 dataset

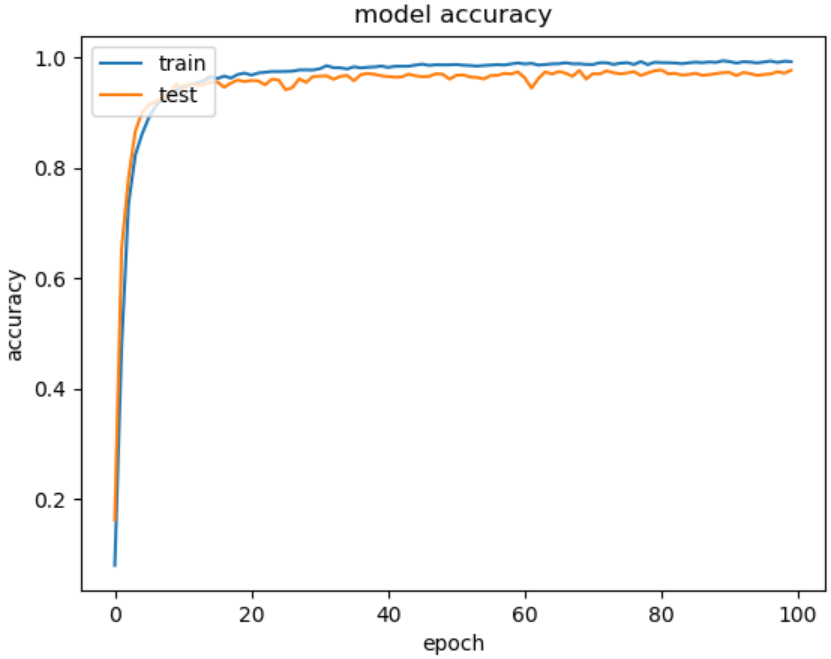


Figure 5.8: Train and test accuracy with respect to no. of epochs for CMATERdb 3.1.2 dataset

Character	P	R	F1	Character	P	R	F1	Character	P	R	F1	Character	P	R	F1	Character	P	R	F1
অ	.97	1.0	.98	ঔ	.94	1.0	.97	ঞ	1.0	.97	.98	ন	.98	.80	.88	ষ	.91	.97	.94
আ	1.0	.97	.98	ক	.94	.97	.95	ট	.97	.97	.97	প	.91	.98	.94	স	.96	.92	.94
ই	.98	.93	.96	খ	.91	.98	.94	ঠ	.95	.97	.96	ফ	.98	.95	.97	হ	.97	1.0	.98
ঈ	1.0	1.0	1.0	গ	.95	.97	.96	ড	.94	.98	.96	ব	.98	1.0	.99	ড়	.98	1.0	.99
উ	.93	.95	.94	ঘ	.96	.90	.93	ঢ	.97	.95	.96	ভ	.97	.93	.95	ঢ়	.97	.98	.98
ঊ	1.0	.97	.98	ঙ	.95	.93	.94	ণ	.83	.95	.88	ম	.94	.98	.96	য়	.92	.98	.95
ঋ	1.0	.98	.99	চ	.97	.98	.98	ত	.98	.97	.97	য	.91	.83	.87	ৎ	1.0	.98	.99
এ	.98	.97	.97	ছ	.97	1.0	.98	থ	.95	.90	.92	র	1.0	.97	.98	ং	.98	1.0	.99
ঐ	.97	.95	.96	জ	1.0	1.0	1.0	দ	1.0	.98	.99	ল	.95	.97	.96	ঃ	1.0	1.0	1.0
ও	1.0	1.0	1.0	ঝ	.98	1.0	.99	ধ	.98	.95	.97	শ	.98	.93	.96	ঁ	1.0	.98	.99

Figure 5.9: Experimental results for each classes of CMATERdb 3.1.2. (Here, P = Precision, R = Recall and F1 = F1-score)

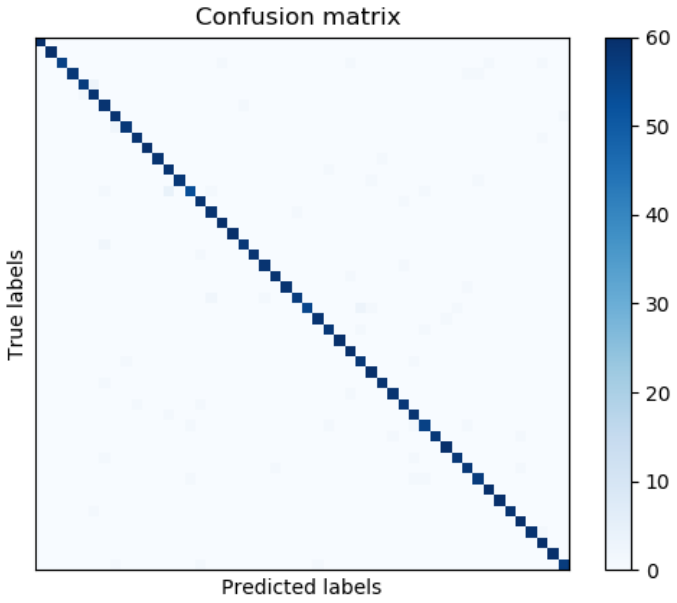


Figure 5.10: Confusion matrix for CMATERdb 3.1.2 dataset

average F1 score is 0.98 all out of 1 which is excellent result considering the complexity of character's shapes and large number of classes. In the confusion matrix, for some classes, lighter color indicates some miss classified data. Similarly, we can see from Fig. 5.9, some classes have lower F1 scores than other. Some of these characters with a relatively more error rate is given in Fig. 5.11 with the number of labels.

Actual Class	Predicted Class
ঘ (15)	খ (13)
ণ (26)	ন (31)
থ (28)	য (37)
ন (31)	ণ (26)
য (37)	ষ, য় (41, 46)

Figure 5.11: Some wrongly classified classes

The most miss classified class is class number 37 with a F1 score of 0.87 which is miss classified as class 41 and 46. Class 26 and 31 with F1 score of 0.88 has been miss classified as class 31 and 26 respectively. Class 15 and 28 with F1 scores of 0.93 and 0.92 is confused with class 13 and 37 in some cases. If we observe, it can be noted, all the miss classified classes are structurally very similar especially class 26, 31 and class 37, 41 and 46. As handwriting varies human to human, the similarity of these characters makes it even harder to classify. Despite these facts, the model performed excellently. Fig. 5.12

Table 5.6: Comparative Analysis: CMATERdb 3.1.2

Method Name	**Accuracy**
LBP + SVM with linear kernel [23]	86.56%
LGP + SVM with linear kernel [23]	86.56%
CENTRIST + SVM with linear kernel [23]	86.64%
NABP + SVM with linear kernel [23]	93.40%
LTP + SVM with linear kernel[23]	90.34%
LAID + SVM with linear kernel[23]	82.16%
DTCTH + SVM with linear kernel [23]	93.24%
Rahman et al. [24]	85.96%
Majid et al. [47]	92.87%
LeNet-5 [48]	89.20%
BanglaNet-14	**97.67%**

and Table 5.6 gives a comparison of proposed technique with some

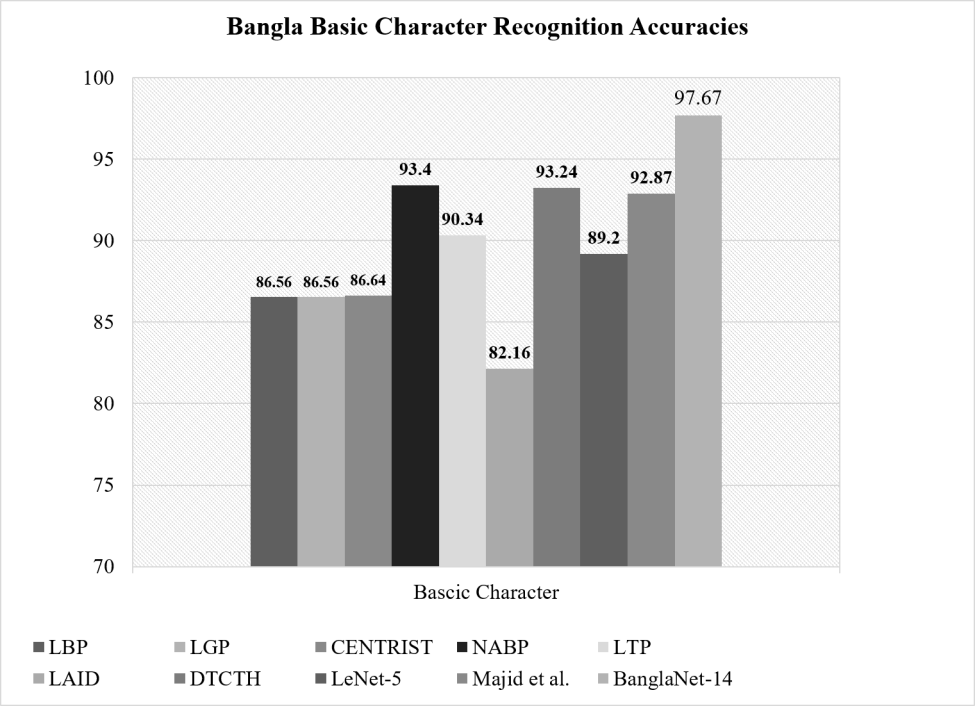


Figure 5.12: Comparison with other methods

other prominent techniques which shows that proposed BBCNet-15 (97.67%) outperforms all other methods. Here LBP, LGP, NABP, CENTRIST, LTP, LAID, DTCTH techniques have been implemented using the same experimental setting as in [23] (details in Chapter 3). In [23] and [47] CMATERdb 3.1.2 dataset was used and in [24] a DCNN model with a customized dataset was used.

5.10 Result and Discussion: CMATERdb 3.1.3.1

Experimental settings for compound character recognition is similar to basic character's. The model was trained for 100 epochs with a learning rate of 0.0005. Fig.5.13 shows the loss function of the model with respect to number of epochs. As can be seen, from the figure, both curves for training and validation loss is close to zero which indicates a properly trained model.To be exact, training loss obtained was 0.1379 and testing loss obtained was 0.0422.

Fig. 5.14 (pg. 65) gives the curve for accuracy obtained for training and validation data. Curves for both types are overlapping indicating a generalized model. Highest training accuracy obtained was 98.67% and validation accuracy was 96.33%. Fig. 5.15 (pg. 66) gives the confusion matrix obtained for all these classes. Using the values of confusion matrix, precision, recall and F1 score of each classes are

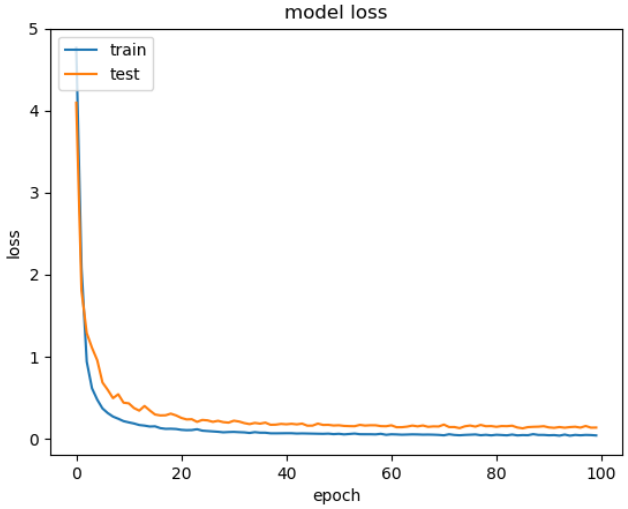


Figure 5.13: Train and test loss with respect to no.of epochs for CMATERdb 3.1.3.1

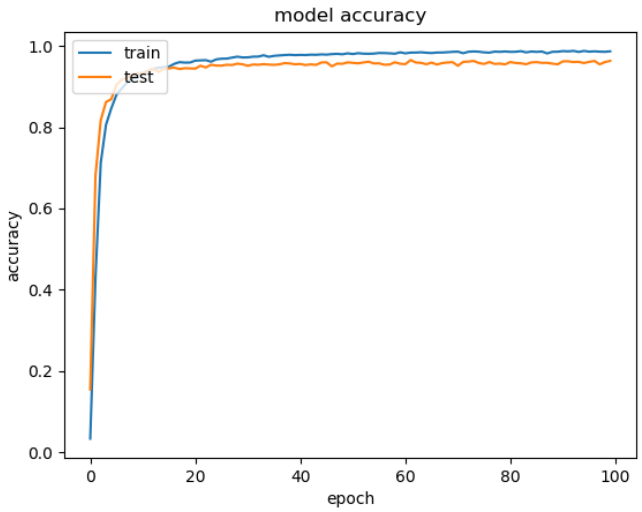


Figure 5.14: Train and test accuracy with respect to no. of epochs for CMATERdb 3.1.3.1

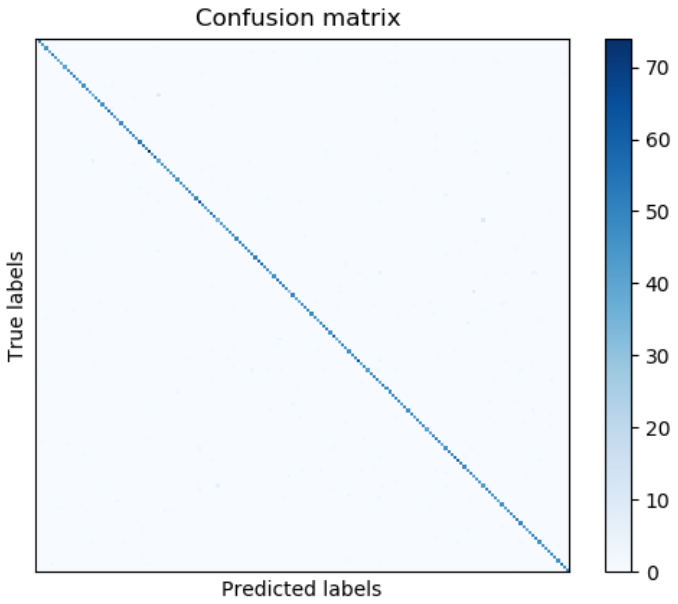


Figure 5.15: Confusion matrix for CMATERdb 3.1.3.1 dataset

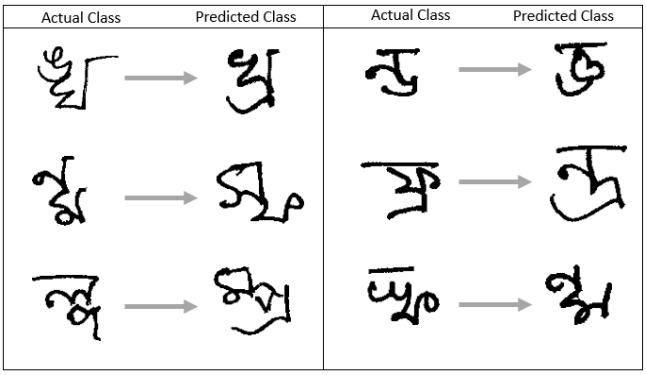


Figure 5.16: Some frequently miss-classified classes

computed for each classes. Fig. 5.16 on page 66 shows some frequently miss classified classes. All characters begin curly shaped and same characters having different representation have made the task for classifier problematic.However, average precision of 0.96, recall of 0.96 and F1 score of 0.96 was obtained which is superior than any existing method. Fig. 5.17 (pg. 67) and Table 5.7 gives a comparative analysis

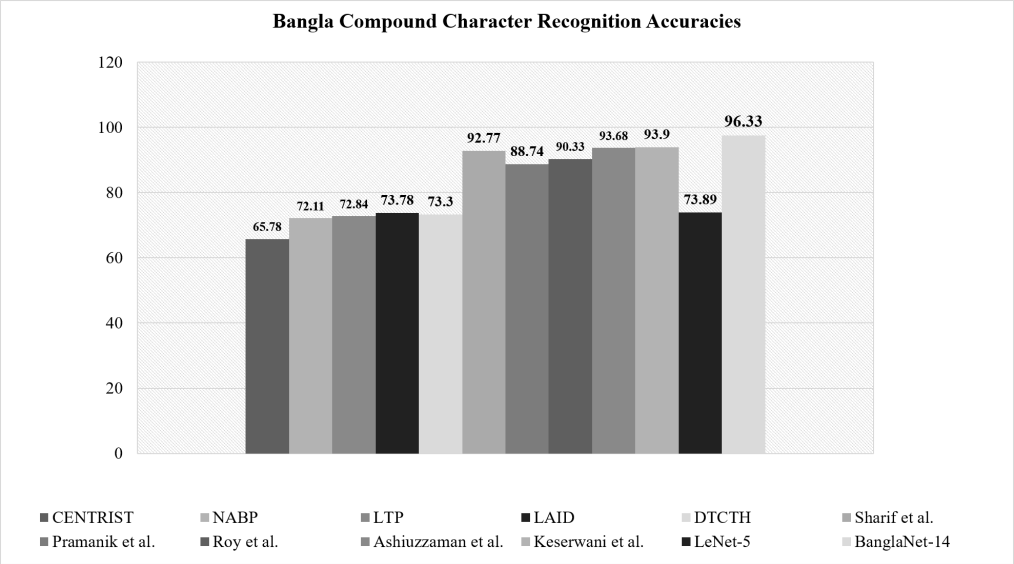


Figure 5.17: Comparative Analysis: CMATERdb 3.1.3.1

of the proposed model with some existing techniques. As can be seen clearly the proposed method outperforms other existing techniques.

Table 5.7: Comparative Analysis: CMATERdb 3.1.3.1

Method Name	**Accuracy**
LBP + SVM with linear kernel [23]	65.69%
LGP + SVM with linear kernel [23]	63.17%
CENTRIST + SVM with linear kernel [23]	65.78%
NABP + SVM with linear kernel [23]	72.11%
LTP + SVM with linear kernel[23]	72.84%
LAID + SVM with linear kernel[23]	73.78%
DTCTH + SVM with linear kernel [23]	73.3%
Sharif et al. [34]	92.77%
Pramanik et al. [33]	88.74%
Roy et al. [29]	90.33%
Ashiquzzaman et al. [31]	93.68%
Keserwani et al. [31]	93.9%
LeNet-5 [48]	73.89%
BanglaNet-14	**96.33%**

Here, parameter settings for LBP, LGP, CENTRIST, NABP, LTP, LAID and DTCTH is described in Chapter 3. Other methods uses CMATERdb 3.1.3.3 which is same as CMATERdb 3.1.3.1 except that, the later is RGB images and first one contains gray scale images. Colors does not really effect handwritten recognition. So, both these datasets can be considered similar. [34, 33, 29, 31, 32] are described in Chapter 2.

5.11 Result and Discussion: CMATERdb 3.1.1

For this dataset same parameter settings as the two other datasets is used. Fig. 5.18 shows the graph for loss functions for both training and validation data. The lowest loss obtained for training is 7.4963^{-4} and for validation data is 0.0577.

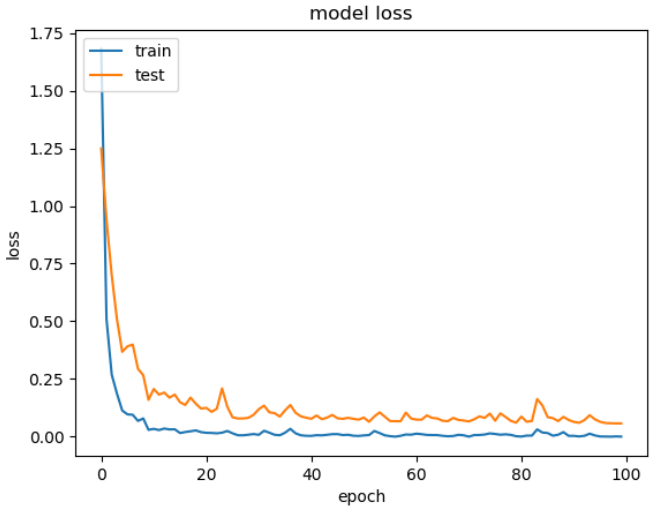


Figure 5.18: Train and test loss with respect to no.of epochs for CMATERdb 3.1.1

Fig. 5.19 on page 69 depicts the graph for training and validation accuracy. Maximum training accuracy obtained is 100% and validation accuracy is 98.35%.

A confusion matrix is shown in Fig. 5.20 from which precision, recall and F1 score is computed (Fig. 5.22). From the confusion matrix we can notice some frequently miss predicted classes which are shown in Fig. 5.23 (pg. 70). We can notice the structural similarities between these classes (Fig. 5.21 on page 70). Although printed form of theses characters are quite distinct, but due to variety of style among different individuals makes some characters similar looking. This is true for any

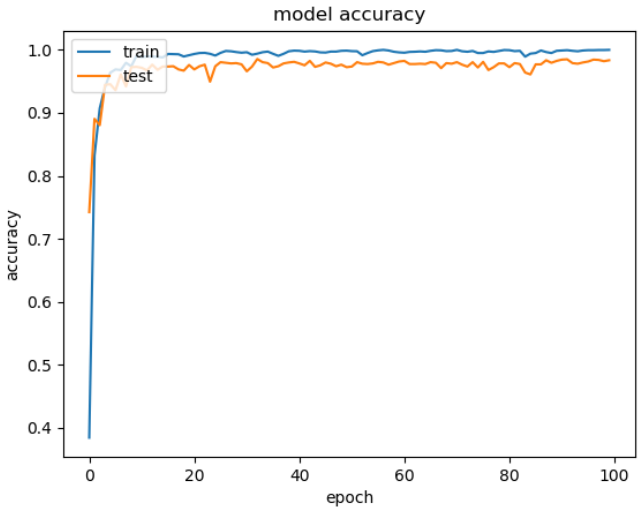


Figure 5.19: Train and test accuracy with respect to no. of epochs for CMATERdb 3.1.1

handwritten data. Despite these problems proposed model performed excellently.

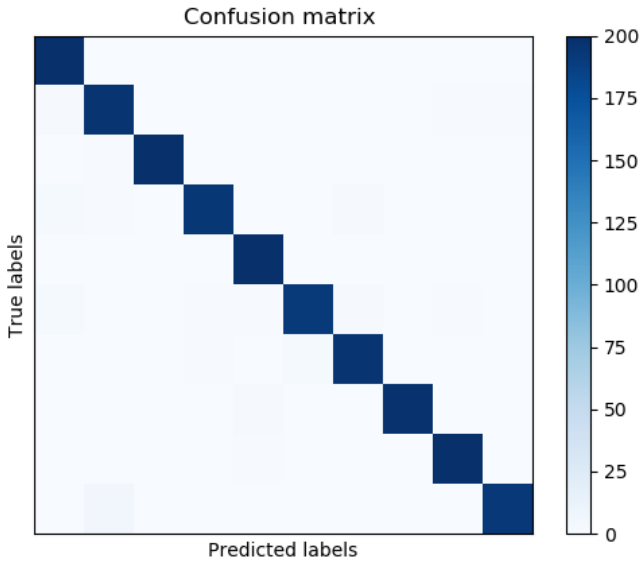


Figure 5.20: Confusion matrix for CMATERdb 3.1.1

Comparative analysis with the proposed model with some existing techniques is shown in Fig. 5.24 and Table 5.8. All of these methods [49, 37, 9, 16] used CMATERdb 3.1.1 dataset. As can be seen evaluation used proposed method is superior then other methods.

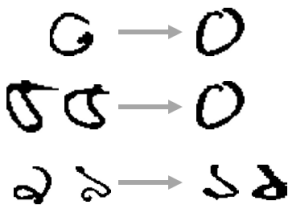


Figure 5.21: Structural similarities

Class	P	R	F1	Class	P	R	F1
০	0.96	1.00	0.98	৫	0.98	0.96	0.97
১	0.96	0.98	0.97	৬	0.98	0.98	0.98
২	1.00	0.99	1.00	৭	1.00	0.99	0.99
৩	0.99	0.97	0.98	৮	0.99	0.99	0.99
৪	0.98	1.00	0.99	৯	0.99	0.96	0.98

Figure 5.22: Experimental results for each classes of CMATERdb 3.1.1. (Here, P = Precision, R = Recall and F1 = F1-score)

Actual Class	Predicted Class
৩	০
৫	০
৯	১

Figure 5.23: Some frquently missclassified digits

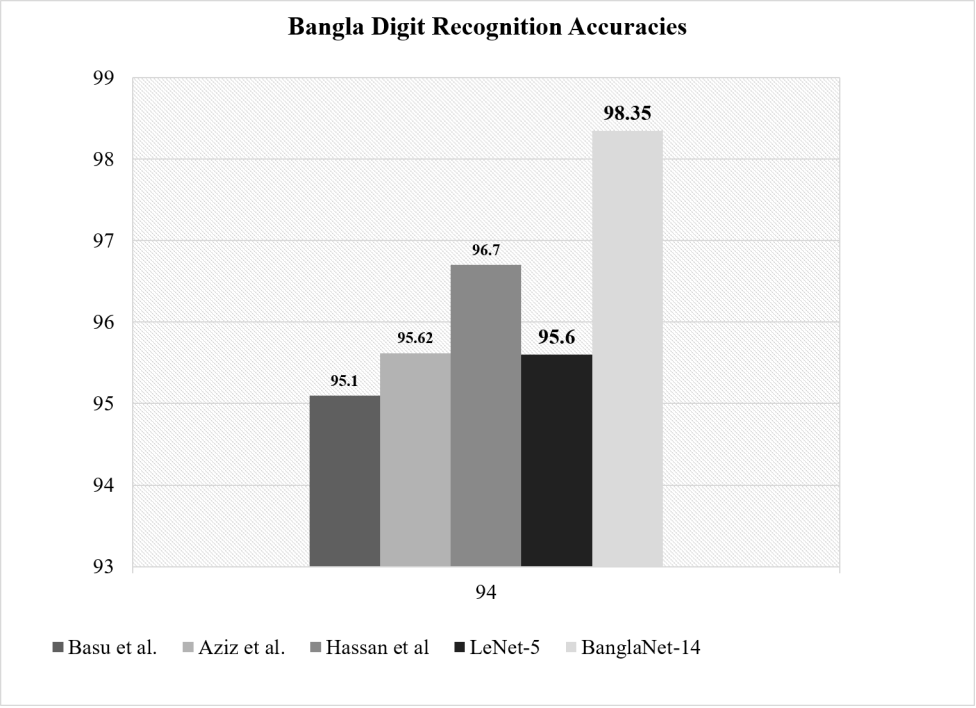
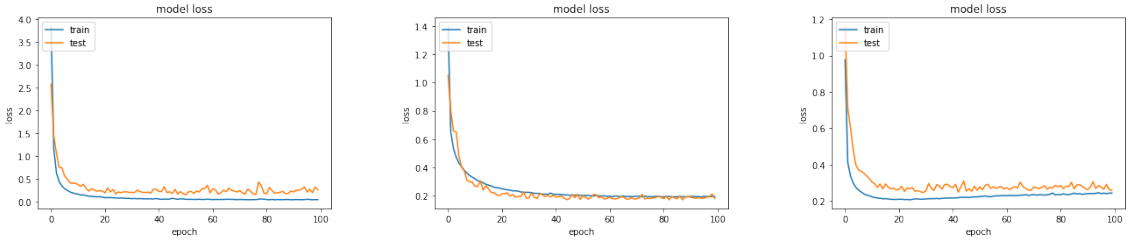


Figure 5.24: Comparative Analysis: CMATERdb 3.1.1

Table 5.8: Comparative Analysis: CMATERdb 3.1.1

Method Name	Accuracy Obtained
Haider Adnan Khan et al. [37]	94%
Basu et al. [49]	95.1%
Aziz et al [9]	95.62%
Hassan et al [16]	96.7%
LeNet-5 [48]	95.60%
BanglaNet-14	**98.35%**

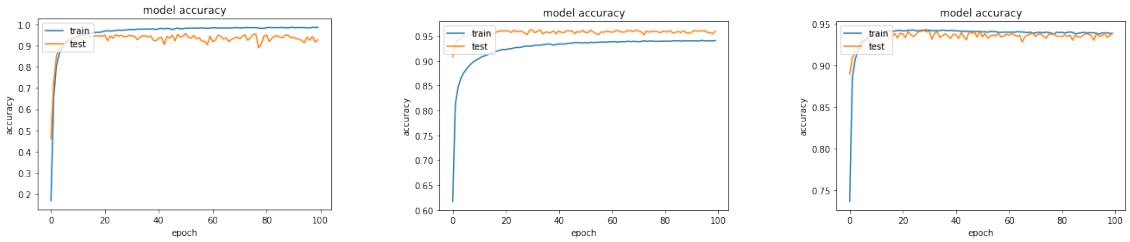


(a) CMATERdb (b) BanglaLekha-Isolated (c) Ekush

Figure 5.25: Train and test loss with respect to no.of epochs for three datasets

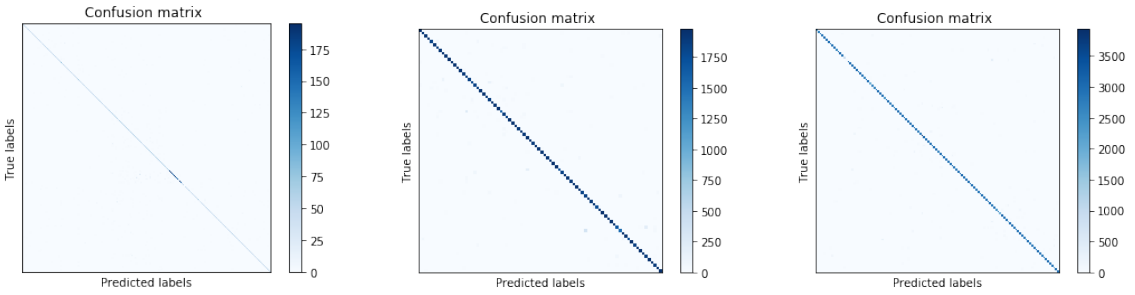
5.12 Result and Discussion: Basic, Compound and Digits Combined

For building an HCR, we need to use a classifier that can detect all Bangla basic characters, compound characters and digits. For this purpose, CMATERdb 3.1.2, CMATERdb 3.1.3.1 and CMATERdb 3.1.1 datasets were combined to form a dataset that contains all Bangla characters and digits. Total 231 number of classes were used to train the model. The model was first trained using LeNet-5 architecture that gives a test accuracy of 79.88% with 1.44 test loss which was not up to the mark. Then, it is trained using BanglaNet-14 which provides highest test accuracy of 95.62% with a loss of 0.1566. Training



(a) CMATERdb (b) BanglaLekha-Isolated (c) Ekush

Figure 5.26: Train and test accuracy with respect to no. of epochs for three datasets



(a) CMATERdb (b) BanglaLekha-Isolated (c) Ekush

Figure 5.27: Confusion matrix for Bangla Characters and Digits for three datasets

accuracy and loss was 98.59% and 0.051 respectively. Fig. 5.25a and 5.26a gives the learning curves for the model.

Two other datsets are used to evaluate the proposed model namely, BanglaLekha-Isolate and Ekush as described in Section 5.2. Training and validation curves obtained for Banglalekha isolated and Ekush dataset is given in Fig. 5.25b and 5.25c. As can be seen, minimum loss obtained for training and test data are 0.1954 and 0.1806 for Banglalekha-isolated dataset. For Ekush dataset it is 0.2430 and 0.2590. Training accuracy obtained for Banglalekha isolated and ekush is 94.06% and 93.87% respectively. Whereas test accuracies are 95.94% and 93.86% as can be seen from Fig. 5.26b and 5.26c. Fig. 5.27a,

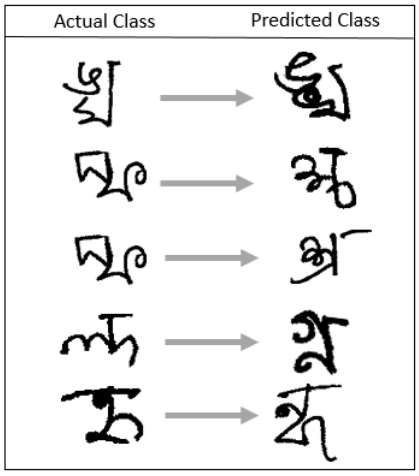


Figure 5.28: Some frequently miss classified characters and digits

5.27b and 5.27c provides a confusion matrices obtained for these three datasets. Average precision, recall and F1-score obtained for 231 classes of CMATERdb dataset is 0.93, 0.92 and 0.92, for 84 classes of Banglalekha isolated is 0.97, 0.97, 0.97 and for 122 classes of Ekush dataset is 0.96, 0.96 and 0.96. Fig. 5.28 gives some frequently miss-predicted classes. As can be seen, all of these wrongly predicted classes

are compound classes. The reason of miss classification is the huge number of classes, very cursive way of writhing compound characters and more than one way of writing same compound characters.

5.13 Summary

From the aforementioned discussion, we can notice that proposed model performed superior then other techniques. The reason of this performance gain is because of the use of layered architecture. As shallower layers of DCNN gains the more absolute features and deeper layers can detect more detailed shapes, which most state-of-art machine learning techniques cannot do, helps DCNN to achieve better results. Using random weights and fitting them according to the problem enables DCNN to efficiently solve any sort of classification problem. Whereas, in state-of-arts machine learning techniques we need to use a separate feature extraction step before the actual classification, a DCNN model extract valuable features while learning. All these valuable and necessary qualities have made DCNN the ultimate and most efficient choice against almost any image classification task.

Chapter 6

Conclusion and Future Works

Handwritten character recognition is one of the most challenging area of image classification for its numerous application in various fields and paying a key role for digitization. Isolated characters recognition is the most important step to build an HCR. Despite a number of research studies, an appropriate and effective model is lacking for classifying all of Bangla handwritten basic characters, compound characters and digits. On account of this, this thesis proposed an effective DCNN model that is shown to provide superior performance. Rigorous experimentation on benchmark datasets for Bangla basic characters, compound characters and digits using state-of-art machine learning methods as well as DCNN models have proven DCNN to be an efficient and effective solution for Bangla isolated characters and digits recognition. Because of its ability to detect micro-structures from an image, extracting features according to the need of model and layered architecture to detect a varied range of features from an image has made DCNN work well with such a difficult problem of handwritten characters' recognition. However, some limitations of the proposed improved DCNN model is:

- All possible compound characters classes are not considered (which is about 300) which can affect the overall performance.
- Using more data can be beneficial to get more accuracte performance in the case of DCNN.

Despite these limitations this model outperforms some significant methods for image classification. Future scope for this thesis includes:

- This research work will be extended to form a complete Bangla HCR.

- Different new techniques of DCNN will be applied on isolated characters to get further performance gain if possible.
- A dataset containing every possible compound characters will be prepared to get a complete solution.

Bibliography

[1] VK Govindan and AP Shivaprasad. "Character recognition—a review". In: *Pattern recognition* 23.7 (1990), pp. 671–683.

[2] Ch N Manisha, E Sreenivasa Reddy, and YK Sundara Krishna. "Role of offline handwritten character recognition system in various applications". In: *International Journal of Computer Applications* 135.2 (2016), pp. 30–33.

[3] Gary F Simons. *Ethnologue: Languages of the world*. sil International, 2017.

[4] Subhadip Basu et al. "Handwritten Bangla alphabet recognition using an MLP based classifier". In: *arXiv preprint arXiv:1203.0882* (2012).

[5] Nibaran Das et al. "An improved feature descriptor for recognition of handwritten Bangla alphabet". In: *arXiv preprint arXiv:1501.05497* (2015).

[6] Nibaran Das et al. "Handwritten Bangla basic and compound character recognition using MLP and SVM classifier". In: *arXiv preprint arXiv:1002.4040* (2010).

[7] Mst Tasnim Pervin, Shyla Afroge, and Aminul Huq. "A feature fusion based optical character recognition of Bangla characters using support vector machine". In: *Electrical Information and Communication Technology (EICT), 2017 3rd International Conference on*. IEEE. 2017, pp. 1–6.

[8] Umapada Pal, Tetsushi Wakabayashi, and Fumitaka Kimura. "Handwritten Bangla compound character recognition using gradient feature". In: *Information Technology,(ICIT 2007). 10th International Conference on*. IEEE. 2007, pp. 208–213.

[9] Talha Ibn Aziz et al. "Bangla handwritten numeral character recognition using directional pattern". In: *Computer and Information Technology (ICCIT), 2017 20th International Conference of*. IEEE. 2017, pp. 1–5.

[10] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 971–987.

[11] Bongjin Jun, Inho Choi, and Daijin Kim. "Local transform features and hybridization for accurate face and human detection." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.6 (2013), pp. 1423–1436.

[12] Md Mostafijur Rahman et al. "DTCTH: a discriminative local pattern descriptor for image classification". In: *EURASIP Journal on Image and Video Processing* 2017.1 (2017), p. 30.

[13] Md Mostafijur Rahman, Shanto Rahman, and Mohammad Shoyaib. "MCCT: a multi-channel complementary census transform for image classification". In: *Signal, Image and Video Processing* 12.2 (2018), pp. 281–289.

[14] Md Mostafijur Rahman et al. "Noise adaptive binary pattern for face image analysis". In: *Computer and Information Technology (ICCIT), 2015 18th International Conference on*. IEEE. 2015, pp. 390–395.

[15] Xiaoyang Tan and Bill Triggs. "Enhanced local texture feature sets for face recognition under difficult lighting conditions". In: *IEEE transactions on image processing* 19.6 (2010), pp. 1635–1650.

[16] Tasnuva Hassan and Haider Adnan Khan. "Handwritten bangla numeral recognition using local binary pattern". In: *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*. IEEE. 2015, pp. 1–4.

[17] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[19] Chandrika Saha, Rahat Hossain Faisal, and Md Mostafijur Rahman. "Bangla Handwritten Digit Recognition Using an Improved Deep Convolutional Neural Network Architecture". In: *Int. Conf. on Electrical, Computer and Communication Engineering*. In press. 2019.

[20] 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.

[21] Tapan Kumar Bhowmik et al. "SVM-based hierarchical architectures for handwritten Bangla character recognition". In: *International Journal on Document Analysis and Recognition (IJDAR)* 12.2 (2009), pp. 97–108.

[22] Nabeel Mohammed et al. "BanglaLekha-Isolated". In: *Mendeley Data, v2* (2017).

[23] Chandrika Saha, Rahat Hossain Faisal, and Md Mostafijur Rahman. "Bangla Handwritten Character Recognition Using Local Binary Pattern And Its Variants". In: In press. 2018.

[24] Md Mahbubar Rahman et al. "Bangla handwritten character recognition using convolutional neural network". In: *International Journal of Image, Graphics and Signal Processing (IJIGSP)* 7.8 (2015), pp. 42–49.

[25] AKM Shahariar Azad Rabby et al. "EkushNet: Using Convolutional Neural Network for Bangla Handwritten Recognition". In: *Procedia computer science* 143 (2018), pp. 603–610.

[26] Nibaran Das et al. "Handwritten Bangla Compound character recognition: Potential challenges and probable solution." In: *IICAI*. 2009, pp. 1901–1913.

[27] Jianxin Wu and Jim M Rehg. "CENTRIST: A visual descriptor for scene categorization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2011), pp. 1489–1501.

[28] SM Zahid Ishraque et al. “A local adaptive image descriptor”. In: *New Review of Hypermedia and Multimedia* 19.3-4 (2013), pp. 286–298.

[29] Saikat Roy et al. “Handwritten isolated bangla compound character recognition: A new benchmark using a novel deep learning approach”. In: *Pattern Recognition Letters* 90 (2017), pp. 15–21.

[30] Nibaran Das et al. “A benchmark image database of isolated Bangla handwritten compound characters”. In: *International Journal on Document Analysis and Recognition (IJDAR)* 17.4 (2014), pp. 413–431.

[31] Akm Ashiquzzaman et al. “An efficient method for improving classification accuracy of handwritten Bangla compound characters using DCNN with dropout and ELU”. In: *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE. 2017, pp. 147–152.

[32] Prateek Keserwani, Tofik Ali, and Partha Pratim Roy. “A two phase trained convolutional neural network for handwritten Bangla compound character recognition”. In: *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. IEEE. 2017, pp. 1–6.

[33] Rahul Pramanik and Soumen Bag. “Shape decomposition-based handwritten compound character recognition for Bangla OCR”. In: *Journal of Visual Communication and Image Representation* 50 (2018), pp. 123–134.

[34] SMA Sharif et al. “Classification of bangla compound characters using a hog-cnn hybrid model”. In: *Proceedings of the International Conference on Computing and Communication Systems*. Springer. 2018, pp. 403–411.

[35] U Pal and BB Chaudhuri. “Automatic recognition of unconstrained off-line Bangla handwritten numerals”. In: *Advances in Multimodal Interfaces—ICMI 2000*. Springer, 2000, pp. 371–378.

[36] Umapada Pal, BB Chaudhuri, and Abdel Belaıd. “A complete system for Bangla handwritten numeral recognition”. In: *IETE journal of research* 52.1 (2006), pp. 27–34.

[37] Haider Adnan Khan, Abdullah Al Helal, and Khawza I Ahmed. “Handwritten bangla digit recognition using sparse representation classifier”. In: *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE. 2014, pp. 1–6.

[38] Nibaran Das et al. “A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application”. In: *Applied Soft Computing* 12.5 (2012), pp. 1592–1606.

[39] Nibaran Das et al. “A statistical–topological feature combination for recognition of handwritten numerals”. In: *Applied Soft Computing* 12.8 (2012), pp. 2486–2495.

[40] Md Zahangir Alom et al. “Handwritten bangla digit recognition using deep learning”. In: *arXiv preprint arXiv:1705.02680* (2017).

[41] Mst Tasnim Pervin, Shyla Afroge, and Aminul Huq. “A feature fusion based optical character recognition of Bangla characters using support vector machine”. In: *Electrical Information and Communication Technology (EICT), 2017 3rd International Conference on*. IEEE. 2017, pp. 1–6.

[42] Jianxin Wu and Jim M Rehg. "CENTRIST: A visual descriptor for scene categorization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2011), pp. 1489–1501.

[43] Ram Sarkar et al. "CMATERdb1: a database of unconstrained handwritten Bangla and Bangla–English mixed script document image". In: *International Journal on Document Analysis and Recognition (IJDAR)* 15.1 (2012), pp. 71–83.

[44] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[45] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[46] Martın Abadi et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[47] Nishatul Majid. "Handwriting Recognition of Bangla and Similar Scripts". In: (2018).

[48] Yann LeCun et al. "LeNet-5, convolutional neural networks". In: *URL: http://yann. lecun. com/exdb/lenet* (2015), p. 20.

[49] Subhadip Basu et al. "Handwritten Bangla digit recognition using classifier combination through DS technique". In: *International Conference on Pattern Recognition and Machine Intelligence*. Springer. 2005, pp. 236–241.

Publications

- "Bangla Handwritten Digit Recognition using an Improved Deep Convolutional Neural Network Architecture," in International Conference on Electrical, Computer and Communication Engineering, 2019.(Published)
  link: https://ieeexplore.ieee.org/abstract/document/8679309