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

A Deep Convolutional Neural Network Based Approach To Recognize Handwritten Bangla Words

Reyad Hasan Rinkon

Reg. No.: 2017331034

4^{th} year, 2^{nd} Semester

Umme Habiba

Reg. No.: 2017331055

4^{th} year, 2^{nd} Semester

Department of Computer Science and Engineering

Supervisor

Enamul Hassan

Assistant Professor

Department of Computer Science and Engineering

7^{th} March, 2023

A Deep Convolutional Neural Network Based Approach To Recognize Handwritten Bangla Words

A Thesis submitted to the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

By

Reyad Hasan Rinkon
Reg. No.: 2017331034
4^{th} year, 2^{nd} Semester

Umme Habiba
Reg. No.: 2017331055
4^{th} year, 2^{nd} Semester

Department of Computer Science and Engineering

Supervisor

Enamul Hassan
Assistant Professor
Department of Computer Science and Engineering

7^{th} March, 2023

Recommendation Letter from Thesis/Project Supervisor

The thesis/project entitled *A Deep Convolutional Neural Network Based Approach To Recognize Handwritten Bangla Words* submitted by the students

1. Reyad Hasan Rinkon
2. Umme Habiba

is under my supervision. I, hereby, agree that the thesis/project can be submitted for examination.

Signature of the Supervisor:

Name of the Supervisor: Enamul Hassan

Date: 7th March, 2023

Certificate of Acceptance of the Thesis/Project

The thesis/project entitled *A Deep Convolutional Neural Network Based Approach To Recognize Handwritten Bangla Words* submitted by the students

1. Reyad Hasan Rinkon
2. Umme Habiba

on 7th March, 2023, hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

Head of the Dept.	Chairman, Exam Committee	Supervisor
Md Masum	Md Masum	Enamul Hassan
Professor	Professor	Assistant Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering	Department of Computer Science and Engineering

Abstract

Handwritten word recognition is the transformation of handwritten text into machine-readable text. It is vital for several applications, including the digitization of historical documents, the processing of automated forms, and customized handwriting recognition. Handwritten texts are more difficult to recognize than printed ones. The size and form of a handwritten text written by a variety of individuals vary. Numerous variations in writing styles complicate the character recognition mechanism. The recognition of Bangla handwritten text has previously been the subject of several scholars. Recent advancements in the domains of image-based identification, computer vision, and natural language processing have been made possible by the convolutional neural network's (CNN) unique ability to extract and classify features. The outcome of this study is a deep convolutional neural network (DCNN)-based method for detecting Bengali handwritten text(characters and words). In the area of pattern recognition, a deep, efficient architecture that can analyze a vast quantity of data is one of the most effective approaches to achieving better accuracy or a less error rate. Therefore, DCNN was employed for both feature extraction and classification in this task. This research evaluates its accuracy by applying the DCNN model to three datasets for character-level work: BanglaLekha-Isolated, CMATERdb, and Ekush data. By using DCNN, 98.78% accuracy has been attained in character recognition. For word-level work we also analyzed another three datasets, Banglawriting, zilla-64, and our custom dataset. Test accuracy of our model for word recognition is 95.78%

Keywords: Handwritten Word Recognition, Handwritten Bangla Word Recognition, Neural Network, Convolutional Neural Network

Acknowledgements

We would like to express our heartfelt gratitude to our thesis supervisor, Enamul Hassan, for his constant support, guidance, and expertise throughout our research journey. His vast knowledge and practical insights enabled us to work with extensive datasets and cutting-edge methodologies. We are thankful to him for providing us with the opportunity to conduct research under his mentorship. His unwavering commitment to our success and his encouragement and inspiration have been invaluable to us. We truly appreciate his guidance and mentorship during this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Handwritten text recognition is a critical task in the field of pattern recognition and machine learning. It has a wide range of applications in various fields, including education, healthcare, and finance. However, recognizing handwritten text accurately is a complex problem due to variations in handwriting styles, quality of images, and noise in the data. Bangla is one of the most widely spoken languages in the world, and it has a unique script that adds to the complexity of the recognition problem. The recognition of Bangla script is a challenging task due to its complex structure and the presence of many ligatures and diacritical marks. The unique features of Bangla script make it difficult to recognize handwritten text accurately, especially in the case of cursive handwriting. Therefore, the development of an efficient and accurate recognition system for Bangla handwriting is highly desirable.

1.1 Motivation

The motivation for doing OCR for handwritten Bangla characters and words is to improve the accessibility and processing of Bangla language data. The Bangla language is widely spoken in Bangladesh, India, and other countries, and its script is known for its complexity and variation. Despite this, there is a lack of resources available for recognizing handwritten Bangla text. This makes it challenging to accurately digitize and process Bangla language data, which is increasingly important in the digital age.

OCR technology has the potential to overcome these challenges by enabling the conversion

of handwritten Bangla text into digital form, which can be processed, searched, and analyzed more efficiently. This can benefit a wide range of industries and applications, such as digitizing historical documents, facilitating data entry in industries that require the processing of Bangla text, and improving accessibility for those with visual impairments.

Furthermore, OCR for handwritten Bangla characters and words can contribute to the development of language technology and natural language processing. Developing accurate OCR algorithms that can handle the complexity and variation of the Bangla script can help unlock the potential of this language for use in various applications.

However, the benefits of developing accurate OCR systems for handwritten Bangla text are numerous. In addition to enabling the digitization and processing of historical documents and other Bangla language data, OCR can also be used for data entry in industries such as banking, healthcare, and education. This can improve the efficiency and accuracy of data processing, reduce errors and costs, and facilitate faster decision-making. Additionally, OCR can improve accessibility for individuals with visual impairments by converting handwritten Bangla text into audio or braille format.

In summary, OCR for handwritten Bangla characters and words has the potential to provide significant benefits for a wide range of industries and applications. While there are challenges that must be overcome, the development of accurate OCR algorithms for Bangla script can help unlock the potential of this language for use in various applications and contribute to the development of language technology and natural language processing.

1.2 Problem Definition

The character list for the Bangla language is quite diverse. Bangla language contains 50 basic characters. There are 11 vowels, and each one of them may take on one of two distinct forms, depending on whether it is used alone or in combination with a consonant in a word. There are also some letters known as compound characters, that are actually combinations of two or more letters. In addition, there are a few characters that seem to be duplicates of one another. Because of these characteristics, it is challenging to figure out the features that are used to recognize bangla texts.

When it comes to handwriting, recognition becomes more challenging. What someone writes

depends on the time or place. Different individuals have their own ways of writing. Even writings by the same person can be drastically different. As well as writing by hand makes it harder to identify the difference between characters that appear alike. Due to this, each of these behavioursÂ makes it harder to figure out handwritten texts.

The problem we are addressing is the recognition of handwritten Bangla words. The dataset we will be using consists of images of handwritten Bangla words, which can be of varying quality and size. The task is to train a model that can accurately recognize the words and classify them into their respective categories.

The specific challenges of the problem are:

- Dealing with the complex structure and unique features of the Bangla script
- Handling the variability of handwriting styles and quality of images
- Overcoming the curse of dimensionality due to the large number of possible input features
- Optimizing the hyperparameters of the CNN to achieve the best performance

1.3 Objectives

The primary objective of our work on OCR for Bangla script at the word and character level is to develop a system that can accurately recognize and digitize handwritten Bangla words and characters . Specifically, we aim to address the challenges posed by the complexity and variation of the Bangla script, as well as the lack of high-quality datasets for training and testing OCR systems.Many machine learning approaches, including ANN, MLP, SVM have been employed to address these handwritten character recognition challenges by many researchers . We plan to achieve this objective by exploring convolutional neural networks (CNNs). Convolutional neural networks are now commonly employed in this field of study. Because this approach extracts the characteristics from the provided input on its own. To get the best possible results from this strategy, we must appropriately pre-process the training data. We intend to employ our deep convolution neural network for character recognition on a significant public dataset (Ekush)[1] that contains 122 classes and of 367,018 isolated characters.Another two public datasets named CMATERdb

3.1.2 [2] and Bangla Lekha Isolated [3] were also used. And for word level recognition we explored publicly available BanglaWriting[4], zill-64 [5] and our custom dataset.

To achieve this objective, we have the following specific goals:

- Preprocessing the dataset to remove noise, resize images, and standardize the format
- Designing and implementing a convolutional neural network (CNN) for recognizing handwritten Bangla words
- Optimizing the network by fine-tuning hyperparameters and employing suitable activation and optimization functions
- Evaluating the performance of the proposed method on benchmark datasets and comparing it with other state-of-the-art methods

1.4 Outline

The remainder of the document is structured in the following manner:

Section 2 : This section provides an overview of related work in the field of handwritten text recognition, with a focus on Bangla script.

Section 3 : This section aims to give a brief overview of the relevant literature and technical knowledge related to the study. Its purpose is to help the reader understand the technical concepts and tools used in the proposed model..

Section 4 : The following section describes the dataset utilized in the study and the measures taken to preprocess the data.

Section 5 : This section describes the proposed convolutional neural network-based model in detail, including the architecture, training process, and optimization techniques used.

Section 6 : This section presents the experimental results and a comparative analysis with other methods.

Section 7 : Finally, we conclude the paper in this with a summary of our findings and future research directions.

Chapter 2

Related Work

The purpose of this chapter is to provide a complete literature analysis of the several methods to handwritten word recognition, covering both standard and deep learning-based techniques. The objective of this chapter is to offer a comprehensive overview of the existing research and to identify the gaps and limits of the present methodologies, which will aid in the development of a unique strategy that overcomes the existing constraints and achieves superior performance.

2.1 Handwritten Character recognition

2.1.1 ANN , SVM and MLP Based Researches

Nibaran Das et al. (2014) came up with a method for classifying 50 basic characters and 10 numbers using 125 features based on a convex hull. For classification, an MLP with one hidden layer was chosen. This method correctly recognized handwritten basic characters 76.86% of the time and numbers 99.45% of the time[6].

Nibaran Das et al. (2009) also suggested using another set of features with 132 features. The feature set had modified shadow features, a quad tree-based longest run feature, distance-based features, and octant and centroid features. In both cases, MLP was used to sort things into groups. The accuracy of 50 basic character classes went up from 75.05% to 85.40% [7].

T.K. Bhowmik et al. proposed a method based on SVM in 2009. The method first groups the image and then finds the actual class, using a three-stage, two-level, hierarchical architecture that performed better than SVM alone. [8].

K. L. Kabir et al. (2015) proposed a method for classification where various projection-based features like left projection features, right projection features, quadratic feature algorithms were experimented using Quad tree-based features, longest run features, and Octant centroid features, and then classified with ANN.[9].

2.1.2 CNN based Researches

A CNN-based technique for BHCR was proposed by Md. Mahbubar Rahman et al. (2015). 50 handwritten Bangla basic characters were classified using two convolutional layers with a 5*5 kernel, two subsampling layers with a 2*2 average area, input and output layers. Each class had 50 neurons in the output layer and 784 neurons in the input layer. Test accuracy was reported to be 85.36% and training accuracy to be 94.55%. [10]

M. A. R. Alif and colleagues (2017) utilized a Resnet architecture for handwritten Bangla recognition characters. Adam optimizer and dropout layers were employed to improve the efficiency of the current Resnet architecture.[11]

A layer-wise training deep neural network method was put out by Saikot Roy et al. (2017) for classifying complex Bangla handwritten characters. They also suggested using the faster-converging RMSProp optimization technique.[12]

To classify 171 composite character classes, A. Ashiquzzaman et al. (2017) suggested a CNN technique that uses dropouts to reduce overfitting andÂ ELU filter to reduce gradient vanishing.CMATERdb 3.1.3.3 served as their database. They recognized 171 different compound characters with 93.68% accuracy.[13]

For the recognition of 122 Bangla handwritten character classes, Akm Shahariar Azad Rabby et al. (2018) developed a CNN architecture with 22 layers. They employed CMATERdb for cross validation and Ekushdb for testing and training. Reports of accuracyÂ 95.01% and 97.73% forÂ CMATERdb and Ekushdb respectively.[1]

A 13-layered CNN with dropout layers was proposed by Akm Shahariar Azad Rabby et al. (2018). Layers with dropouts were utilized to minimize overfitting. This model also makes use of the Adam optimizer. They used three separate databases to test this model.[14]

In order to categorize 171 compound characters, A. Fardous et al. (2019) developed a CNN architecture made up of 8 convolutional layers, 4 pooling layers, 2 fully connected layers, and Relu

activation function. Dropout reduced overfitting and relu introduced non linearity. This technique has a reported accuracy of 95.5%.[15]

Authors	Method	Dataset and Classes	Accuracy
Nibaran Das [6]	MLP	1000 sample of 50 class	Train:85.40%
T.K Bhowmik[8]	MLP and SVM	27000 sample and 45 class	Train:89.22
Nibaran Das [7]	MLP and SVM	19776 and 50 class	MLP:79% SVM:80%
K. L. Kabir[9]	ANN	ISI Dataset , 60 Characters	Train:84.14%
R. Sarkhel[16]	SVM	CMATERdb, 231 characters	Basic:86.53% , compunt:78.38%, Mixed:72.85%
Mahbubar Rahman[10]	CNN	20000, 50 basic	Train:94.55% Test:85.36%
M. A. R. Alif[11]	CNN(Modified ResNet)	Isolated Bangla(84 classes), CMATERdb(231 Classes)	Proposed ResNet1895.10% ResNet18:94.52% ResNet34:94.59% Test:85.36%
A. Ashiquzzaman[13]	CNN	CMATERdb, 171 class	Testing 93.68%
saikot Roy [17]	DCNN	CMATERdb , 171 Classes	Train:90.68%
A K M Shahriar[1]	CNN	Ekush(122 Class) CMATERdb(171 Class)	EkushDb:97.73% CMATERdb:95.01%
A K M Shahriar[14]	CNN	CMATERdb and ISI(50 Basic)	ISI:95.7% CMATERdb:98%
Md Zahangir[18]	DCNN	CMATERdb ,171 Classes	Basic:98.31% Numerals:99.13%
Sourajit Saha[19]	DCNN	Bangla Lekha Isolated(84 Classes)	Test:97.21%
A. Fardous[15]	CNN	CMATERdb(171 character)	Test:95.5%
Md. Zahidul Islam[20]	CNN	Ekush ,122 Class	Train:90%
Mr Moynuddin [21]	CNN	Ekush ,122 Class	Train:98.68%
Mr Moynuddin [21]	CNN	Bangla Lekha Isolated(84 Classes)	Train:92.67%

Table 2.1: Literature Review Summery for character Recognition.

2.2 Printed Word Recognition

A study[23] published in 1998 where Chaudhuri presents a comprehensive Optical Character Recognition (OCR) system Bangla. It is the first OCR system for any subcontinental language. The proposed system is susceptible to skew correction, text and graphic separation, line segmentation, zone identification, and word and character segmentation. For single font clear texts, 95.50% word level recognition accuracy has been achieved, which is comparable to 99.10% character level recognition accuracy. To combine Bangla words, the authors used a technique known as "diphone concatenation" and "utterance rules" to convert graphemes into phonemes.

Another study[24] by Satter provides an innovative strategy for detecting and segmenting printed Bengali characters. There is no need to normalize the component since it is independent of size. Several printed Bangla phrases or textual images were used to test and monitor the outputs of the system. In 99.50% of occurrences, isolated characters fall into an isolated group, while the bulk of errors result from connected characters and characters with many points of contact. To examine the system's segmentation skills, distorted text patterns or textual pictures have been used. In 2011, Miah presented this[25] study and proposed an Artificial Neural Network-based strategy for identifying Bangla words on traffic signs. It employs feature extraction methods such as rotation, translation, and scaling invariant feature extraction to obtain the statistical and momentary characteristics of the word. A multilayer neural network using back propagation learning is used to recognize the whole word. Using a variety of word kinds, the throughput is evaluated, and the accuracy rate is found to be 92.13 percent.

2.3 Handwritten Word Recognition

2.3.1 Online

A technique to online Bangla handwriting identification is proposed here[26], one of the first to examine cursively written words rather than individual letters. It is based on hidden Markov models and combines a sub-stroke level feature representation of the Bangla script with an HMM-based writing model that employs context-dependent sub-word units. And the percentage of identification for holistic Bengali words is 88%. A framework for real-time recognition of handwritten Bangla words using a set of fuzzy linguistic criteria is presented in this[27] research. They collected 500

samples of handwritten words from 10 participants and discovered that the framework properly recognizes 386 words, with an accuracy rate of 77%. The identification performance is dependent on word length and writing style, and the segmentation procedure must be enhanced to discover and eliminate irrelevant points and embrace more fuzzy features.

This[28] article presents a method for segmenting Bangla words based on a technique for establishing busy zones at the stroke level. It was examined on 6,500 online handwritten Bangla word samples with a segmentation accuracy of 98.45 percent. Their suggested technique can support almost all Bangla word forms, from horizontally aligned to multidirectionally skewed.

2.3.2 Offline

The authors of this[29] study presented a fuzzy technique for segmenting handwritten Bangla word pictures in 2007. Their segmentation success rate averages 95.32 percent. The approach is capable of handling 50 basic letters, 10 modified shapes, and 250 compound characters with complex formed complex shapes in the Bangla script. The authors believe it might be used for the segmentation of handwritten Bangla words.

This[30] work presents a stroke-based lexicon reduction strategy for reducing the search area for handwritten word recognition. The concept requires two properties of a word picture to form a feature vector: the word's length and its shape. The experiment was conducted using 35,700 offline photographs of handwritten Bangla words.

In 2014, Bhowmik and others provides a method for recognizing handwritten Bangla words [31]. Histograms of Oriented Gradients (HOG) are utilized as the feature set to represent each word sample in the feature space, and a neural network-based classifier is employed to categorize the samples. The performance on a short dataset is 87.35 percent, and the suggested approach works well despite the complexity and abundance of the Bangla script.

In the next year, another study[32] by Bhowmik follows the trend of holistic word recognition for bangle handwritten words. They have used concentric rectangles and convex hull-based characteristics to categorize word pictures belonging to various classes. 2754 handwritten Bangla word samples are gathered from various sources, and a neural network-based classifier is selected. The recognition performance of the methodology is examined using a three-fold cross-validation method, and in the best case scenario, the suggested system successfully recognizes 84.74 percent

of word pictures.

An all-encompassing approach for recognizing handwritten Bangla words is proposed in this[33] paper. From handwritten word pictures, a collection of elliptical features is derived to represent them in the feature space. Five well-known classifiers are compared in order to pick the optimal classifier. A neural network-based classifier is selected for the recognition challenge. Using a limited dataset, the suggested system's precision is 85.88%.

This[34] study provides a horizontally segmented Indic handwritten word recognition system (upper,middle and lower). This method decreases the number of separate component classes and improves system recognition. Water reservoir-based characteristics have improved zone segmentation and character border identification, and a new sliding window-based feature termed Pyramid Histogram of Oriented Gradient (PHOG) has been presented for middle zone classification. This research provides a HMM-based segmentation-free method for effective Indic handwritten word recognition. The suggested method divides words into three zones and recognizes them zone-wise. This study got 83.39% accuracy.

A comprehensive approach for recognizing handwritten Bangla city names is described in this[35] article. The algorithm attained an accuracy of 90.65 percent on 10,000 samples consisting of the 20 most prevalent city names in West Bengal, India. From each of the grids, gradient-based features were retrieved, inspired by the Histogram of Oriented Gradients (HOG) feature descriptor. The recognition accuracies of five well-known classifiers were examined, and the classifier Sequential Minimal Optimization (SMO) was selected based on its superior performance.

The study cited as [36] introduces a hierarchical feature selection (HFS) model based on Genetic Algorithm (GA) to maximize the local and global features extracted from each of the handwritten word images considered. The dataset used contained 80 city names, with 150 samples per city. The proposed model achieved an accuracy of 95.30%.

In 2018, a study proposed a method for recognizing 80 handwritten Bangla city names using a holistic-word recognition approach [37]. The method involves creating a new shape-based feature vector of size 186 for each word image and testing it on a database of 12,000 handwritten word images. The proposed method achieved a recognition accuracy of 87.50%. However, because the size of the feature vector is large, it takes more time to build the train module.

In 2018, Bhowmik introduced a comprehensive approach for recognizing handwritten words by

merging multiple elliptical, tetragonal, and vertical pixel density histogram-based characteristics here[38]. Using fivefold cross-validation, the suggested system outperforms with SVM than MLP for the provided dataset, attaining 83.64% accuracy in the best case and 79.38% accuracy on average with SVM.

This[39] paper by Dutta provides an end-to-end trainable CNN-RNN hybrid architecture for Devanagari and Bangla.

In 2020, an article[40] presented H-WordNet, a deep convolutional neural network-based holistic technique for recognizing handwritten words. The model consists of four convolutional layers and one fully connected layer to efficiently categorize word pictures, and the effectiveness of various pooling techniques with the proposed model is studied. The findings reveal that the H-WordNet achieves 96.17% recognition rate.

Hossain presents this[41] paper where a segmentation and recognition method is proposed for identifying and extracting handwritten Bangla words using Convolutional Neural Network (CNN) The achieved accuracy is 84% for character level testing and 82% for word level testing.

This[42] comparative research, published in 2021, analyzes the efficacy of segmentation-free techniques for detecting handwritten Bangla cursive words. Five commercially available classifiers and CNN-TL architectures are used to extract statistical feature sets from word images. The report concludes by comparing its results to all prior testing using a seven-layer FCN layout. They achieved an accuracy of 98.86% using ResNet50 .

Authors	Method	Dataset and Classes	Accuracy
Dasgupta et al. [43]	SVM	32 classes	Accuracy:76.69%
Tamen et al.[44]	MLP	21 class	Train:96.82%
Bhowmik et al. [45]	HMM+GA	119 classes	MLP:79.12%
Barua et al.[35]	SMO	20 Classes	Train:90.65%
Bhowmik et al.[16]	SVM	120 classes	Basic:83.64%
K. Dutta et al. [39]	CNN	287 classes	Test:87.23 %
D. Das et el. [40]	CNN	120 Classes	Train: 96.17%
M. T. Hossain et el.	CNN [41]	122 Classes	Train: 84%
R. Pramanik et el. [42]	CNN(Resnet-50)	120 Classes	98.68 %
Farisa Benta et el[46].	DenseNet101	73 Char Class	WER: 0.26
Md Ali Azad et el.[5]	DCNN	64 Class	93.30 %

Table 2.2: Summary of Literature Review for character Recognition.

Chapter 3

Background Study

3.1 Convolutional Neural Network

3.1.1 Basic Structure

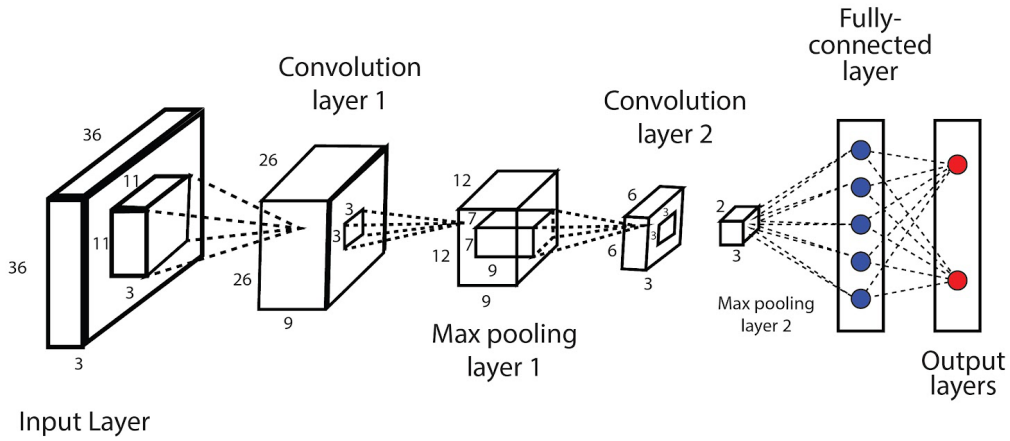

Figure 3.1: Basic Structure of CNN

Convolutional neural networks are a sort of hierarchical model whose input consists of raw

datasets such as RGB, Grayscale or Binary images, raw audio data, etc[47]. Comparing Convolutional Neural Networks to other classification methods, much less pre-processing is required. Convolutional Neural Networks are capable of learning various filters and characteristics, whereas filters in basic systems are hand-engineered after enough training. The structure of a convolutional neural network was influenced by the way the visual cortex is organized and is similar to the connectivity pattern of neurons in the human brain. The Receptive Field, a restricted region of the visual field, is the area where individual neurons are triggered. To fill the entire visible area, several of these fields overlap [48].

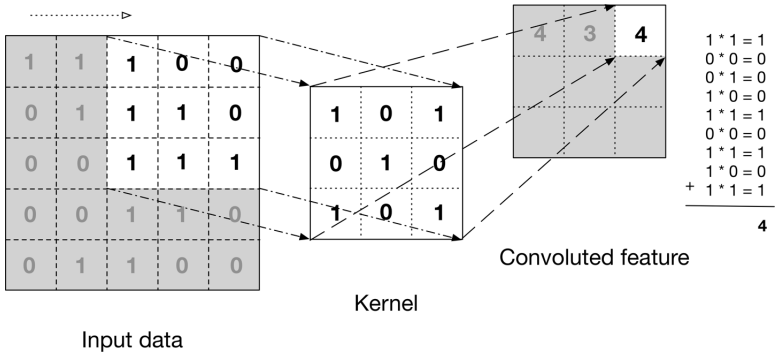

Figure 3.2: Convolutional Layer

3.1.2 Convolutional Layer

The fundamental component of a convolutional neural network (CNN) is the convolutional layer, where most of the computation occurs. This layer requires several inputs, including input data, a filter, and a feature map. Convolution, on the other hand, is a mathematical operation that combines two functions and shows how one function can modify the shape of another function.[49]

$$s(t) = \int x(a)w(t-a)\partial a \tag{3.1}$$

An asterisk is often used to indicate the convolution process:

$$s(t) = (x * w)(t) \tag{3.2}$$

In equation 4.2, the variables x and w stand for the input of a convolutional layer and the kernel, respectively. The created feature map is the result[50].

Assume that the input will be a 3D pixel matrix containing a color image. As a result, the input will have three dimensions corresponding to RGB in a picture: height, width, and depth. In addition, we have a feature detector, also known as a kernel or filter, which traverses the image's receptive fields to determine whether the feature is there.

The feature detector in a convolutional neural network (CNN) is typically represented by a 2-dimensional (2-D) array that is weighted. This array corresponds to a section of the image, and its size is controlled by the filter size, which typically ranges from 2x2 to 3x3 but can vary. The filter is applied to a section of the image, and the dot product between the filter and the input pixels is computed to produce an output array. This process is repeated as the filter moves through the image with a specified stride. The result of this process is a feature map, activation map, or convolved feature that represents the dot products obtained from the input and the filter.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.3}$$

Convolution is commutative so,

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.4}$$

It's worth mentioning that the weights of the feature detector are constant as it slides over the images, a technique known as parameter sharing. During training, the weights may be modified through backpropagation and gradient descent. However, before the neural network is trained, three hyperparameters must be adjusted because they affect the size of the output volume.These are

1. **The number of filters :** The depth of the output is determined by the number of filters. For instance, three distinct filters would result in three different feature maps.[51]

2. **Stride :** The stride of the kernel in a CNN determines the distance it travels across the input matrix in terms of pixels. Although stride values of two or higher are uncommon, a longer stride will lead to a smaller output.[52]

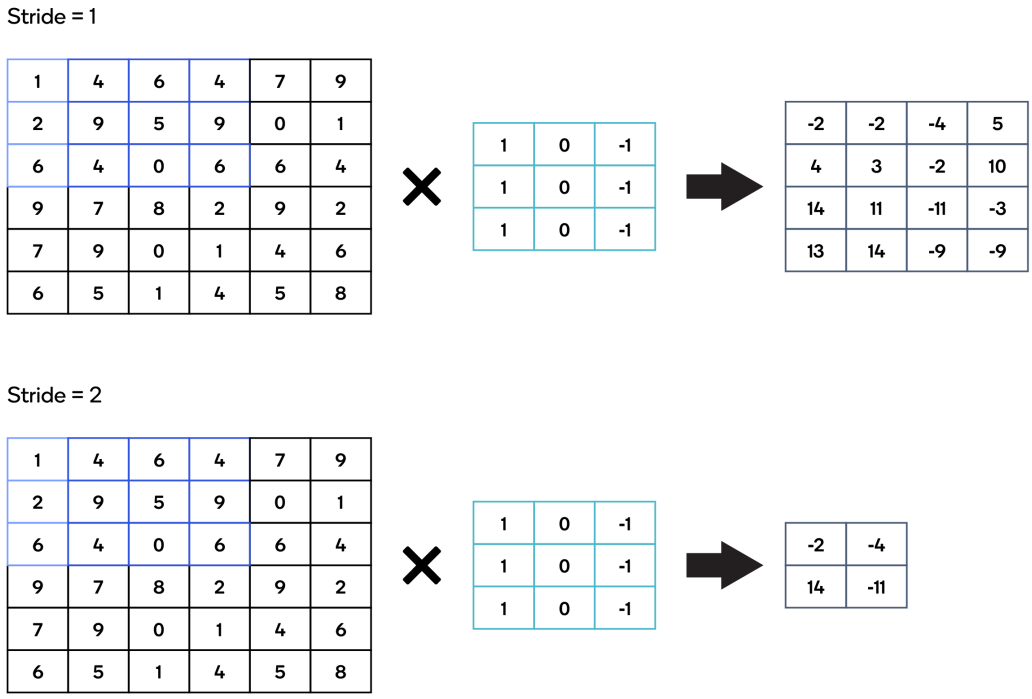

Figure 3.3: Effect of Stride in CNN

3. **padding :**A convolutional neural network uses padding to expand the region of an image it can analyze. Padding is added to the image's frame to provide the kernel additional room to process the image. This helps the kernel process the image more quickly.[3]

   (a) **Same Padding :** In order to enable the filter to cover the edge of the matrix and perform inference on it, padding layers are added in this type of padding. These layers add zero values to the outer border of the images or data.

   (b) **Valid Padding :** Also referred to as no padding, this If the dimensions are out of alignment in this scenario, the last convolution is dropped.

   (c) **Full Padding :** By extending the input's border with zeros, this type of padding makes the output larger.

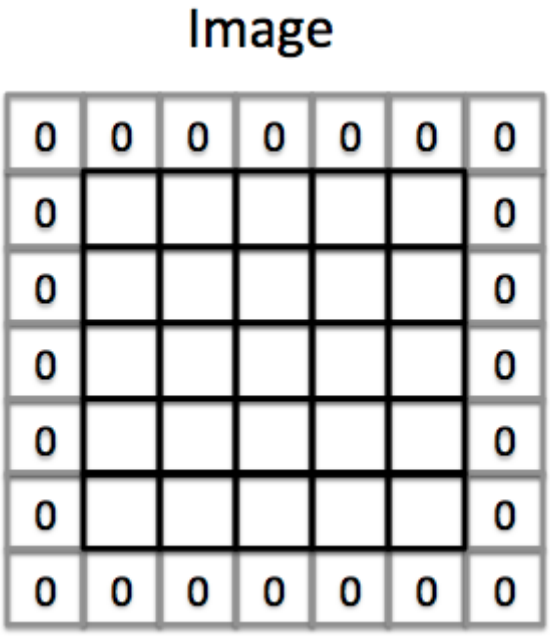

Figure 3.4: Example of Padding

3.1.3 Pooling Layer

Pooling layers, also called downsampling, decrease the number of parameters in the input by performing dimensionality reduction. The pooling operation works similarly to the convolutional layer, in that a filter is applied across the entire input. However, in contrast to the convolutional layer, the filter in pooling layers does not have weights. Instead, the output array is populated by the kernel, which applies an aggregation function to the values in the receptive field.There are mainly two forms of pooling:

1. **Max pooling :** During the maximum pooling process, which is also referred to as max pooling, the maximum value in each patch of each feature map is determined. It's worth noting that this method is more commonly used than average pooling.

$$f(x) = \max(x_{[i,i+k],[j,j+k]}) \tag{3.5}$$

2. **Average pooling :** When using average pooling, the average value is computed for each patch of the feature map. As the filter moves across the input, it determines the average value

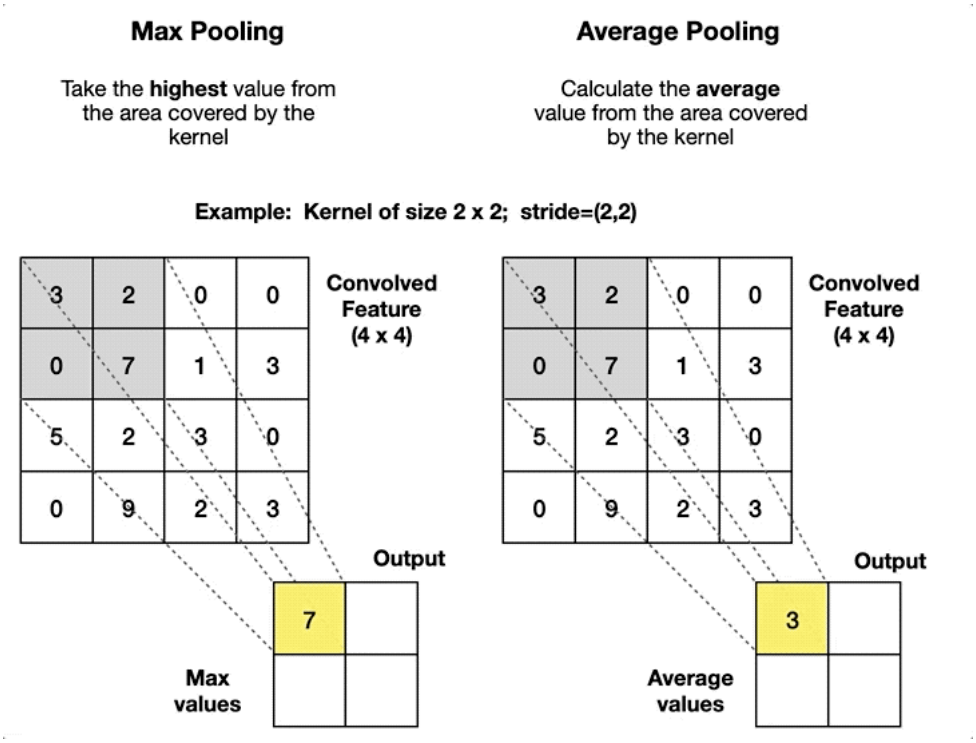

Figure 3.5: Max Pooling and Avarage Pooling

in the receptive field and sends it to the output array.

$$f(x) = \frac{1}{k \times k} \sum_{i=i_l \ldots i_{l+k}} \sum_{j=j_l \ldots j_{l+k}} x_{i,j} \tag{3.6}$$

3.1.4 Activation Function

In a neural network, an activation function is used to convert the weighted sum of the input into an output from one or more nodes in a network layer. The activation function's role is to introduce nonlinearity into the neural network. During forward propagation, an additional step is required for activation functions at each layer, but this computation is considered valuable.

1. **Binary Step :** The binary step activation function is a fundamental function that is often used when we need to limit the output. Essentially, it is a classifier that is based on a threshold, whereby a threshold value is selected to determine whether the output of a neuron should be activated or inhibited.

$$BinaryStep(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

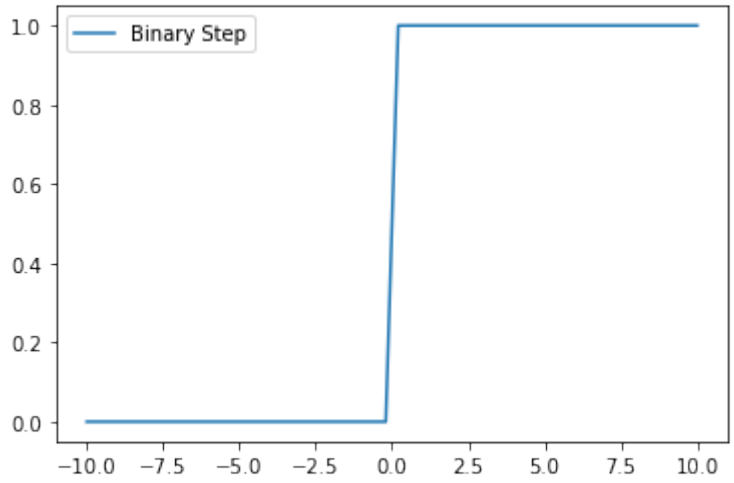

Figure 3.6: Binary Step Activation Function

2. **Rectified Linear Activation (ReLU) :** The rectified linear activation function, known as ReLU, is a function that is linear in parts, outputting the input directly if it is positive, and 0 otherwise. ReLU has become the default activation function in many neural network applications because models trained with ReLU are simpler and typically produce better results than models trained with more complex activation functions.[53]

$$ReLU(x) = max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

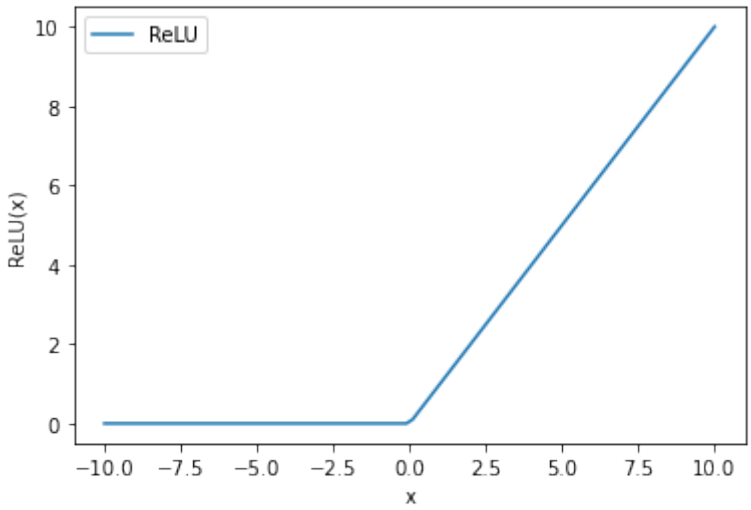

Figure 3.7: ReLU Activation Function

3. **Sigmoid Function :** Sigmoid accepts a real value as input and returns a value between 0 and 1 as output. It is simple to handle and possesses all the desirable qualities of activation functions: non-linearity, continuous differentiation, fixity, and a set output range. [54]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.9}$$

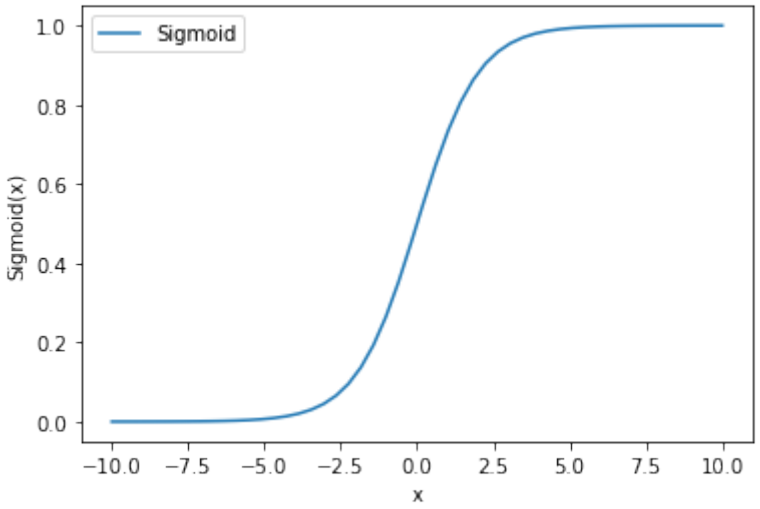

Figure 3.8: Sigmoid Activation Function

4. **Tanh Function :** Tanh activation function is marginally superior to the sigmoid function; like the sigmoid function, it is used to estimate or discriminate between two classes, but it maps only negative inputs to negative quantities and ranges from -1 to 1.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.10}$$

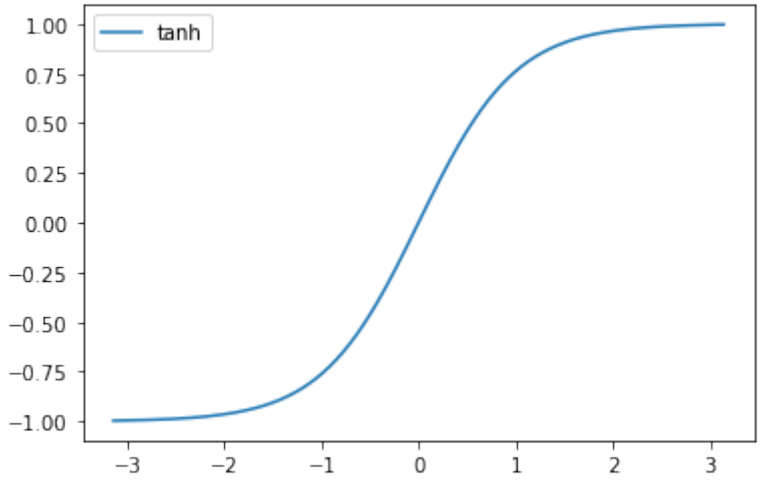

Figure 3.9: Tanh Activation Function

5. **Softmax activation function :** Softmax is mostly utilized in the output layer for decision

making, similar to how sigmoid activation functions.Softmax assigns a value proportional to the input variable's weight, and the sum of these weights is one.

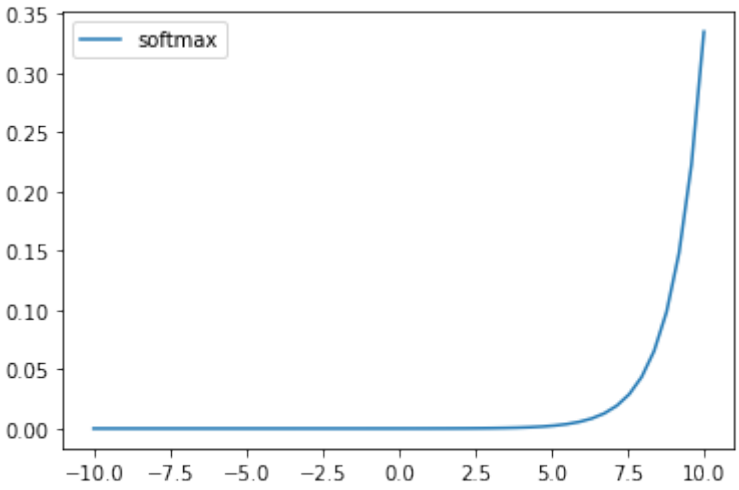

Figure 3.10: Softmax Activation Function

3.1.5 Objective Function

The purpose of the objective function is to measure the difference between the predicted and actual values. In modern convolutional neural networks, the categorical cross-entropy cost function and the L2 loss function are the most commonly used objective functions for classification and regression problems. **Binary Cross Entropy:** Cross-entropy loss, which is also referred to as log loss, is a metric used to evaluate the performance of classifiers that produce probability values between 0 and 1 as output. Cross-entropy loss increases as the predicted probability deviates from the actual label. For instance, if a model predicts a probability of 0.012 when the true observation value is 1, the model would experience a significant loss value. A perfect model would have a log loss of 0.

$$f(x) = \frac{1}{N} \sum_{i}^{N} \sum_{j}^{M} y_{ij} \log(p_{ij}) \tag{3.11}$$

3.1.6 Optimizer Function

In order to improve the performance of our network, we need to use the loss calculated by the loss function. The aim is to minimize the loss as a lower loss indicates a better performing model. Optimization is a mathematical process that aims to minimize or maximize a statement.

Adaptive Moment Optimization One optimization technique is Adaptive Moment Estimation (Adam), which is a form of stochastic gradient descent. It uses adaptive estimation of first-order and second-order moments. Adam is similar to RMSprop and stochastic gradient descent with momentum. It adjusts the learning rate using squared gradients, similar to RMSprop, and it uses a moving gradient average instead of the gradient itself, similar to SGD with momentum, to take advantage of momentum.citezhang2018improved

3.2 Tools

3.2.1 TensorFlow

TensorFlow is a popular, free and open-source software framework. It is widely used for machine learning and artificial intelligence. Within Googleâs Machine Intelligence research or-

ganization, Google Brain Team developed this tool and it was first released in 2015. There is an updated version of tensorflow released in 2019. With TensorFlow, programmers can build dataflow graphsâstructures that depict the flow of data via a graph or a collection of processing nodes. Tensolflow encompasses the whole procedure from beginning to end. It provides an integrated suite of resources for developers, organizations, and academics who are interested in advancing the state of the art in machine learning and create highly scalable applications that are driven by ML. Programming languages as diverse as Python, JavaScript, C++, and Java can all use TensorFlow.

3.2.2 Keras

Keras API was designed to prioritize the user experience rather than machine efficiency. Keras adheres to established standards for minimizing the cognitive burden on users. Its APIs are uniform and easy to use, reducing the need for extensive user input, and it provides clear and prompt error messages. Furthermore, it contains abundant training resources and documentation.

3.2.3 Jupyter Notebook

The Jupyter Notebook is a free, open-source online software for creating and sharing documents with embedded code, mathematical expressions, graphical representations of data, and markdown text. Some of Jupyter's most notable characteristics are listed here. The benefits of using a coding environment that highlights syntax errors, tabs in automatically, and completes tabs. Simply execute the code in the browser and show the output in the box below the code. Whether you're making a remark or a statement in the code, you may do it using the Markdown syntax and modify it right in your browser. the capability to quickly incorporate LaTeX-based mathematical notation into markdown blocks. Rendering the computation results in a visual medium such as HTML, latex, PNG, SVG, etc.

3.2.4 Libraries

We have used Python for this research . There are a bunch of libraries in python. Here is a overview of some of them that we have used in our proposed model.

1. **Numpy** NumPy is a Python library primarily used for handling arrays. It also has capabilities

for dealing with matrices, Fourier transforms, and linear algebra functions. Created by Travis Oliphant in 2005, it is an open source project that is available for free use by anyone.

2. **Pandas** To manage and examine data, the Python programming language has a package called pandas. It has specialized data structures and operations designed for working with computational tables and time-series data. The software is distributed under the three-clause BSD license and is freely available.

3. **Matplotlib** Matplotlib is a Python library for creating visualizations and charts, which can be used with NumPy for numerical computations. It provides an object-oriented application programming interface (API) for embedding plots into programs, and can be used with all-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK.

4. **Scikit-learn** Scikit-learn is a Python machine learning package that provides a range of clustering, classification, and regression methods, such as SVMs, gradient boosting, k-means, random forests, and DBSCAN. It is designed to work seamlessly with other Python libraries like SciPy and NumPy. The scikit-learn project was initiated by David Cournapeau, who developed it as a Google Summer of Code project. It was originally called scikits.learn and was named after "Scikit," another third-party addition to SciPy.

5. **Open CV** OpenCV is an important open-source library that deals with image processing, computer vision, and machine learning. It is widely used for real-time processing in modern systems. With OpenCV, it is possible to analyze images and videos to detect various objects including human handwriting. By combining it with other libraries like NumPy, Python can handle the OpenCV array structure for analysis. We can use vector space and mathematical operations to identify visual patterns and their different features.

Chapter 4

Dataset

4.1 Datasets

4.1.1 Character Level Datasets

1. Ekush:Ekush dataset of isolated Bangla handwritten charactersÂ data was gathered from 3086 people representing university, school, and college students, with approximately 50% 1510 male and 50% 1576 female participants.After writing the handwriting characters on a form, it was scanned to obtain image data from raw data.

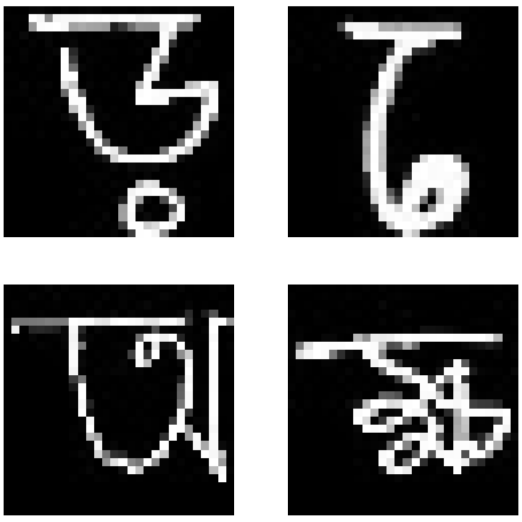

Figure 4.1: Ekush Dataset Example

2. Banla Lekha Isolated:Handwriting examples of all Bangla basic characters and numbers (50 basic characters and 10 numerals) are included in the data. It also includes 24 carefully chosen compound characters. As a result, the dataset contains 84 distinct Bangla characters. Following the collection of raw data on forms, the samples are digitized and pre-processed.

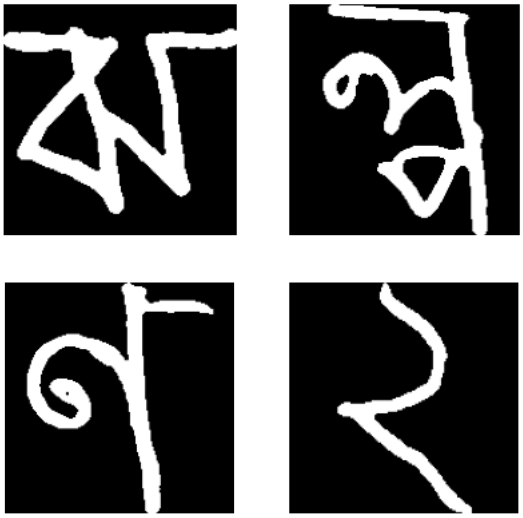

Figure 4.2: Bangla Lekha Isolated Dataset Example

3. CMATERdb 3.1.2: It is a dataset containing 15000 samples of 50 basic bangla character.It is created by CMATER research labratory of Jadavpur University.All the samples here are handwritten images.

4. Combined Dataset: All three datasets are combined together and produced a new dataset. We have ran an experiment using all the data here mentioned

4.1.2 Word Level Datasets

1. BanglaWriting: The BanglaWriting dataset contains 21,234 words contributed by 260 individuals of various ages and personalities. There are 5,470 distinct words and 32,787 characters included. Furthermore included are 450 strikes and errors. All bounding boxes and word levels are generated manually.

2. Zilla-64: Another important Bangla Handwritten Word dataset is Zilla 64. (BHW). It contains the names of all 64 districts of Bangladesh. It has images of 64 classes. Each

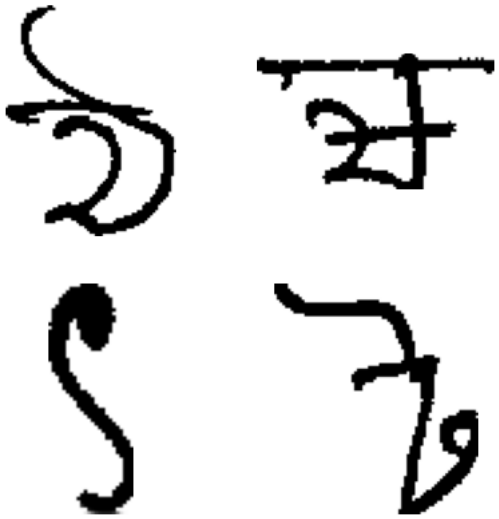

Figure 4.3: CMARTdb 3.1.2 Dataset Example

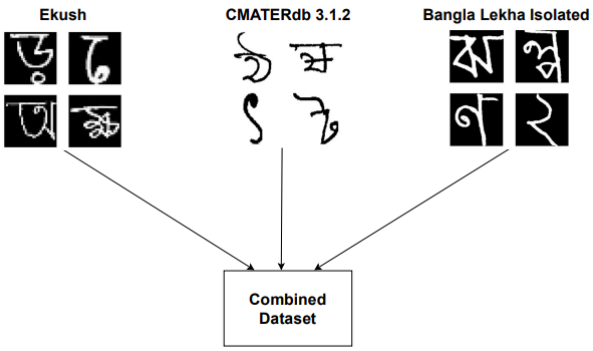

Figure 4.4: Combination of 3 character level dataset

category has roughly 23 photos.

3. Our custom supplementary dataset: To gain more train data we have also collected some word images. We selected these words by frequently analyzing the BanglaWriting dataset. We first sorted all the unique words of the BanglaWriting dataset based on their frequency in descending order. Then we selected from the 751st to 1000th data. And designed 4 different forms each containing 66 boxes for collecting data. We collected 7500 words of 250 classes each class having 30 samples. After preprocessing the scanned images of the data from the bounding box have been cropped and collected in the relevant directory.

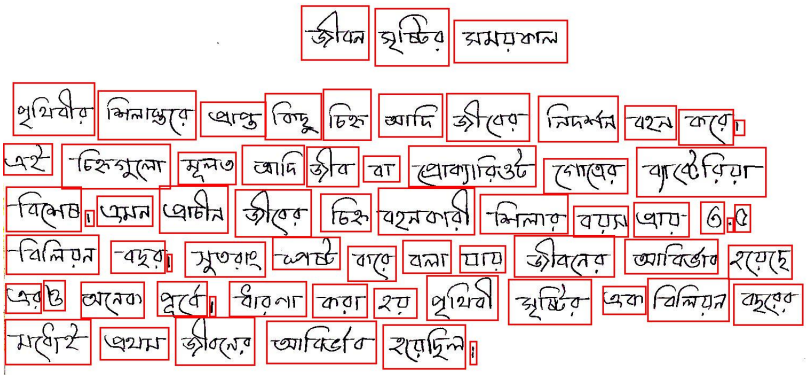

Figure 4.5: BanglaWriting Dataset Example

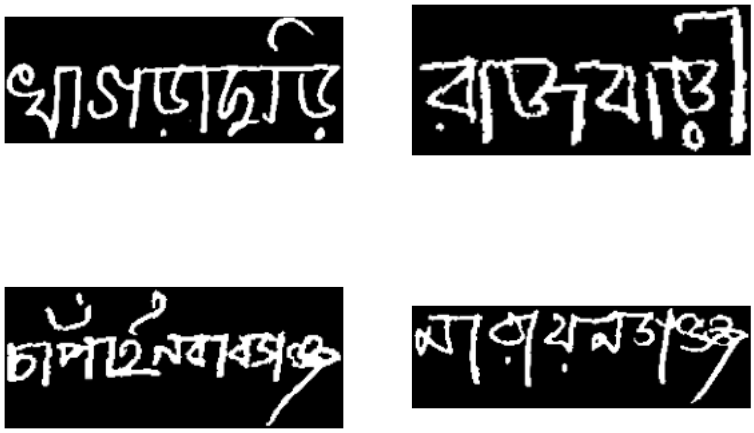

Figure 4.6: Zilla-64 Dataset Example

4.2 Our Custom Word Level Dataset

4.2.1 Purpose of the dataset

The purpose of a custom word-level supplementary dataset is to have a larger number of samples for training and testing a handwriting recognition model. By collecting more samples, the model can learn a wider range of variations in handwriting styles, which can improve its accuracy and generalization performance. This can be especially important in situations where the model needs to recognize handwriting from different regions or languages, or from individuals with unique writing styles. Additionally, having a supplementary dataset can allow for more extensive testing and evaluation of the model's performance, as it can provide a more diverse set of samples to assess its accuracy and robustness.

4.2.2 Data source

The purpose of a custom word-level supplementary dataset is to have a larger number of samples for training and testing a handwriting recognition model. By collecting more samples, the model can learn a wider range of variations in handwriting styles, which can improve its accuracy and generalization performance. This can be especially important in situations where the model needs to recognize handwriting from different regions or languages, or from individuals with unique writing styles. Additionally, having a supplementary dataset can allow for more extensive testing and evaluation of the model's performance, as it can provide a more diverse set of samples to assess its accuracy and robustness.

4.2.3 Designing Collection Methods

To collect the data for the custom dataset, a specific form was designed with designated boxes for writing the handwritten Bangla words. Above each box, a label was provided to indicate the word that needed to be written in the respective box. The forms were distributed among the data sources, which were different individuals between the ages of 19 and 25. The individuals were asked to populate the data by writing the Bangla words in the designated boxes according to the label provided above each box. The process of collecting the data in this way ensured that the data was collected in a structured manner, with each sample being associated with the correct label. Additionally, by collecting data from 91 different individuals, the dataset was diverse and more representative of the variations in handwriting that can be expected in the real world. In total, 7500 words were collected, with 250 classes. This provided sufficient data for training and testing the models.

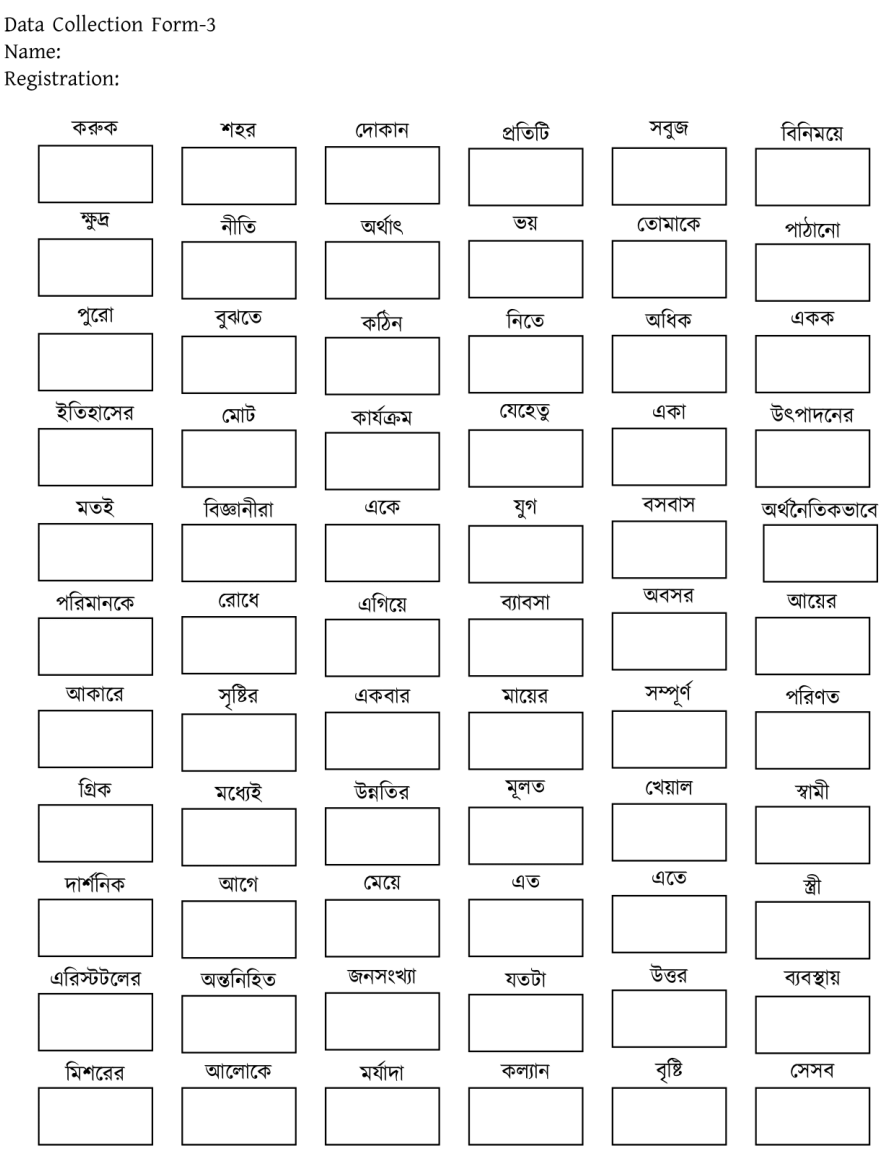

Figure 4.7: Example of Form that was used to collect data

4.2.4 Digitizing, Cleaning and Preprocessing

To digitalize and clean the collected data in paper, the handwritten word images were scanned at a high resolution and saved in a digital format using Canon CanoScan LiDE 300 Flatbed Scanner. Then, image processing techniques such as noise removal, thresholding, and binarization were applied to the scanned images to enhance their quality and extract the handwritten words accurately. Finally, any remaining noise or artifacts were manually removed from the images using image editing software

4.2.5 Annotation

Automatic annotation was performed by utilizing the fact that the boxes in the forms were at the same distance level. A script was developed to automatically detect and extract the word images from the designated boxes. The extracted word images were then processed and cleaned.

Figure 4.8: Example of Form with collected data before annotation

Figure 4.9: Example of Form with collected data after annotation

Chapter 5

Proposed Model

5.1 Models For Character Recognition

5.1.1 Our Proposed Models

In this paper, we propose 2 models of convolutional neural networks for character recognition. One is 11 Layers deep and another is 26 layers deep. Both models are based on convolutional neural network. Each models contain an input layer that accepts 32 by 32 pixel images. Then, a series of 2D convolutional and max pooling layers are used. In model 2, batch normalization has been added after each max pooling layer. Batch normalization is a technique for deep learning that improves the stability and performance of neural networks. It accomplishes this by standardizing the inputs to each layer such that the mean and standard deviation of the inputs are nearly uniform across the training instances within a batch.The total number of parameters varies very much.

No.	Layer(Type)	Output Shape	Total Number of parameters
1	input 1 (InputLayer)	[(None, 32, 32, 3)]	0
2	conv2d (Conv2D)	(None, 32, 32, 32)	320
3	max pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
4	conv2d 1 (Conv2D)	(None, 14, 14, 64)	18496
5	max pooling2d 1 (MaxPooling2D)	(None, 7, 7, 64)	0
6	conv2d 2 (Conv2D)	(None, 5, 5, 128)	73856
7	max pooling2d 2 (MaxPooling 2D)	(None, 2, 2, 128)	0
8	flatten (Flatten)	(None, 512)	0
9	dense (Dense)	(None, 128)	65664
10	dropout 1 (Dropout)	(None, 128)	0
11	dense 1 (Dense)	(None, 122)	15609

Figure 5.1: Architecture of Model 1

No.	Layer(Type)	Output Shape	Total Number of parameters
1	input 1 (InputLayer)	[(None, 32, 32, 3)]	0
2	conv2d (Conv2D)	(None, 32, 32, 32)	896
3	batch normalization	(None, 32, 32, 32)	128
4	conv2d 1 (Conv2D)	(None, 32, 32, 32)	9248
5	batch normalization 1	(None, 32, 32, 32)	128
6	max pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
7	conv2d 2 (Conv2D)	(None, 16, 16, 64)	18496
8	batch normalization 2	(None, 16, 16, 64)	256
9	conv2d 3 (Conv2D)	(None, 16, 16, 64)	36928
10	batch normalization 3	(None, 16, 16, 64)	256
11	max pooling2d 1 (MaxPooling2D)	(None, 8, 8, 64)	0
12	conv2d 4 (Conv2D)	(None, 8, 8, 128)	73856
13	batch normalization 4	(None, 8, 8, 128)	512
14	conv2d 5 (Conv2D)	(None, 8, 8, 128)	147584
15	batch normalization 5	(None, 8, 8, 128)	512
16	max pooling2d 2 (MaxPooling2D)	(None, 4, 4, 128)	0
17	conv2d 6 (Conv2D)	(None, 4, 4, 256)	295168
18	batch normalization 6	(None, 4, 4, 256)	1024
19	conv2d 7 (Conv2D)	(None, 4, 4, 256)	590080
20	batch normalization 7	(None, 4, 4, 256)	1024
21	max pooling2d 3 (MaxPooling2D)	(None, 2, 2, 256)	0
22	flatten (Flatten)	(None, 1024)	0
23	dropout (Dropout)	(None, 1024)	0
24	dense (Dense)	(None, 2048)	2099200
25	dropout 1 (Dropout)	(None, 2048)	0
26	dense 1 (Dense)	(None, 122)	249978

Figure 5.2: Architecture of Model 2

Parameter Name	Parameters for Model 1	Parameters for Model 2
Non-trainable params	0	1,920
Trainable params	173,945	3,523,354
Total params	173,945	3,525,274

Table 5.1: Parameter Information of Proposed Model 1 and Model 2

5.1.2 Classic Vision Models And Transfer Learning

Transfer learning involves utilizing feature representations from a previously-trained model so that a new model does not need to be trained from start.

Pre-trained models are typically trained on enormous datasets that serve as a benchmark at the frontlines of computer vision. The weights derived from the models are reusable for other computer vision applications.

we have tested the datasets using classical computer vision models through transfer learning namely Resnet-50, VGG16, and MobileNet.

- Resnet-50
- VGG16
- MobileNet

Model	Weights	Total params	Trainable params	Non-trainable params
resnet50	imagenet	28,034,042	4,446,330	23,587,712
vgg16	imagenet	16,015,290	1,300,602	14,714,688
mobilenet1.00	imagenet	5,578,042	2,349,178	3,228,864

Table 5.2: Comparison of transfer learning models

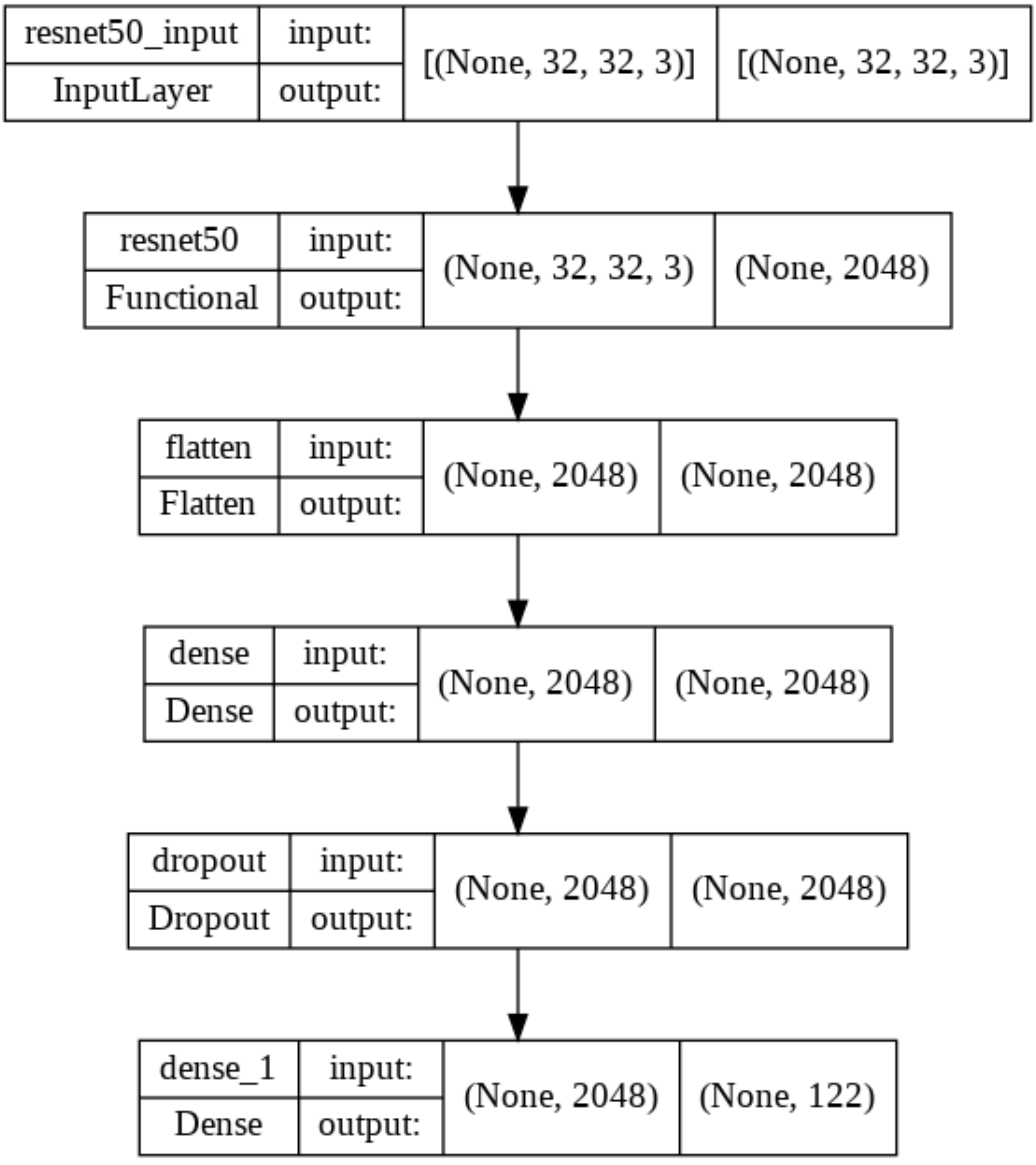

Figure 5.3: Resnet50 Architecture Using Transfer Learning

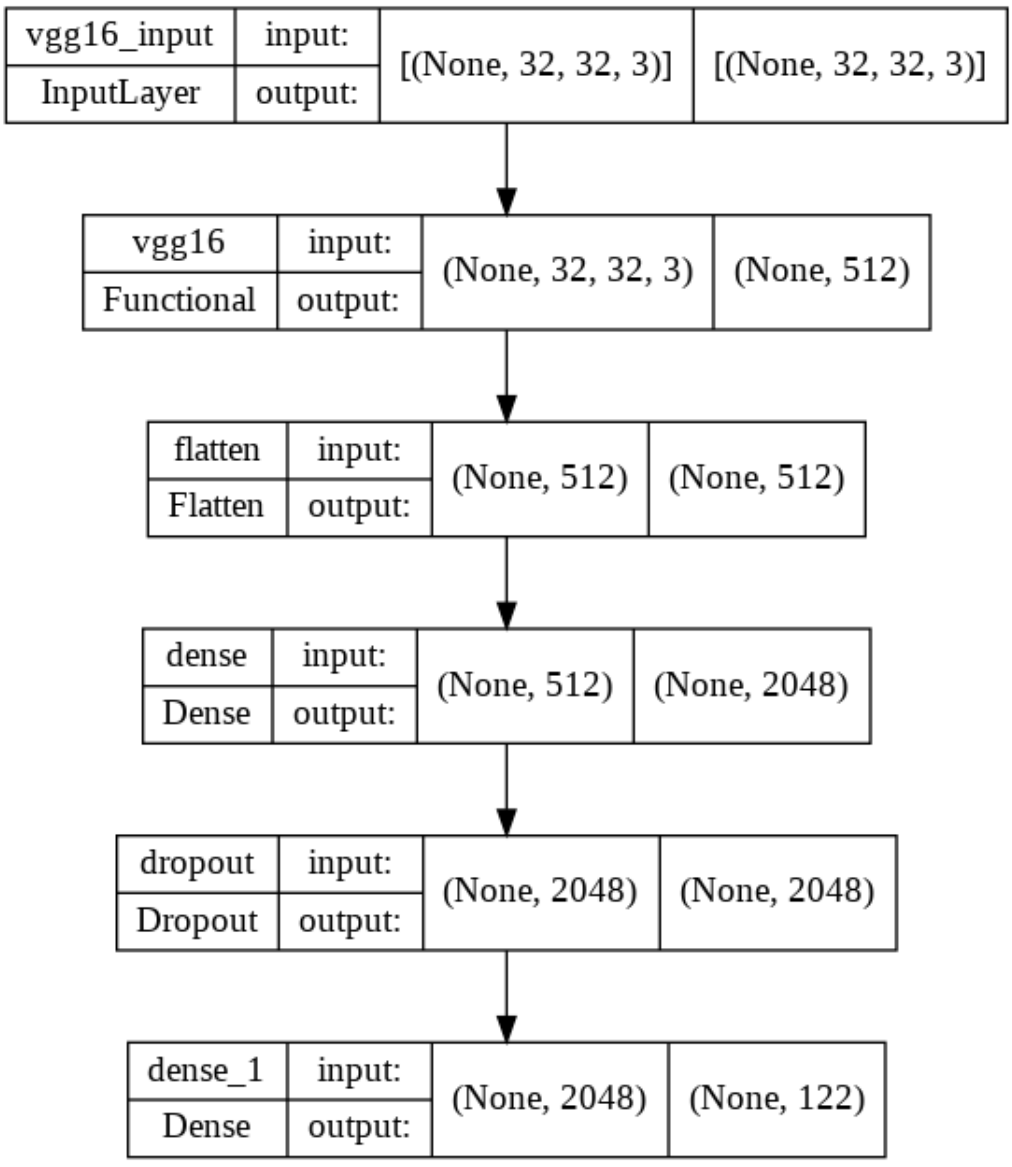

Figure 5.4: VGG-16 Architecture Using Transfer Learning

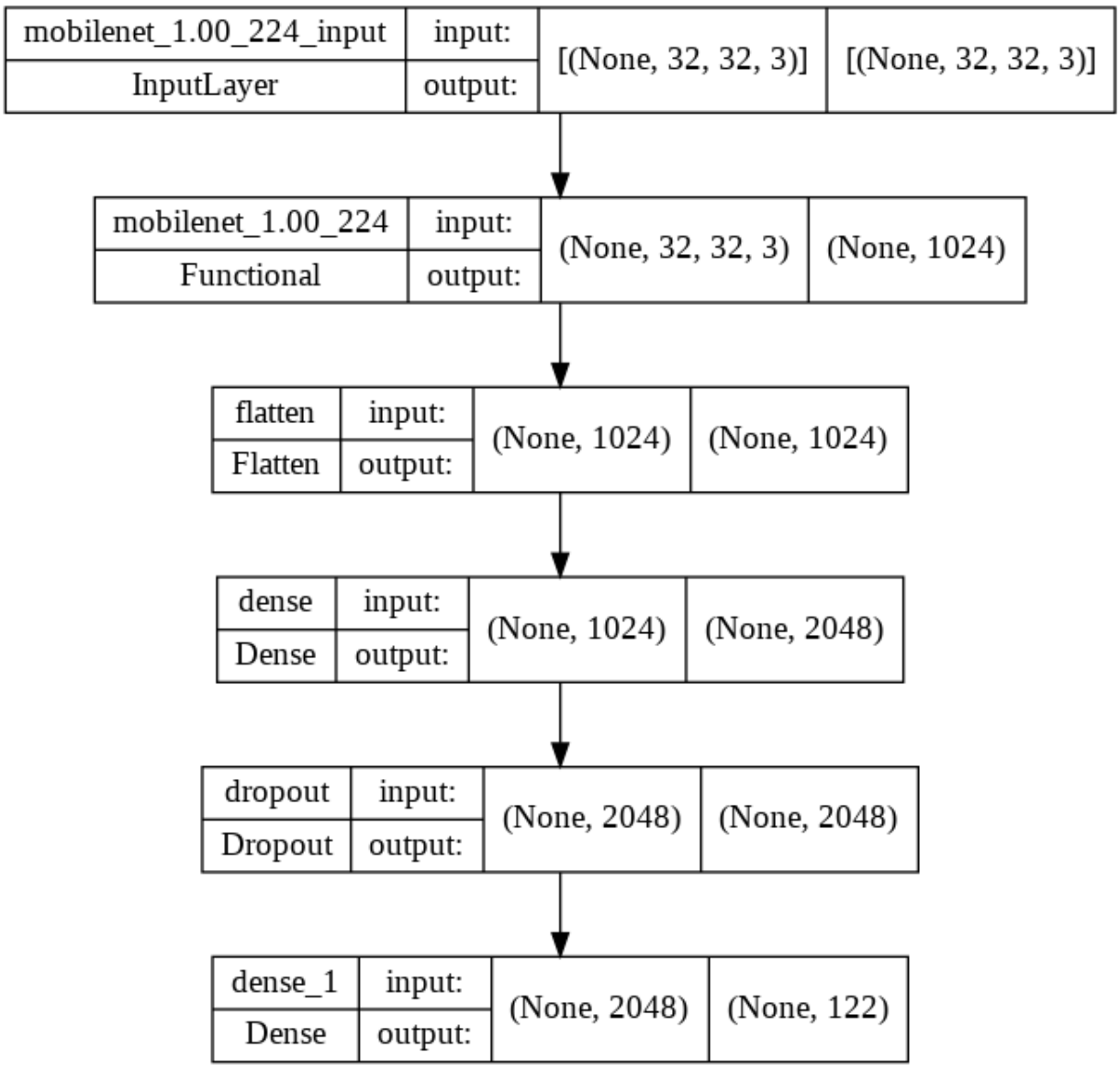

Figure 5.5: MobileNet Architecture Using Transfer Learning

5.2 Models for Word Recognition

5.2.1 Our Proposed Model

We propose model 3 for the purpose of classifying words. It is an architecture using convolutional neural networks. It is supposed to be extremely profound and efficient, while maintaining a high degree of accuracy in picture classification tasks.

Similar to GoogleNet, this architecture employs "Inception modules," which are multi-scale convolutional filters that enable the network to capture features at various scales.These modules are designed to perform a variety of operations simultaneously, including 1x1 and 3x3 convolutions, as well as pooling operations.

This architecture's usage of auxiliary classifiers is a further significant feature. These classifiers are added to the network at intermediate layers to enable it to learn from the extracted features at these layers. This can prevent the issue of vanishing gradients, which occurs when gradients become too small to update the network's parameters efficiently.

No.	Layer(Type)	Params	No.	Layer(Type)	Params
1	input1 (InputLayer)	0	35	conv2d10 (Conv2D)	9264
2	conv2d (Conv2D)	2688	36	concatenate3	0
3	batchnormalization	384	37	conv2d11 (Conv2D)	20064
4	activation (Activation)	0	38	conv2d12 (Conv2D)	119872
5	conv2d1 (Conv2D)	3104	39	batchnormalization8	384
6	conv2d2 (Conv2D)	27680	40	batchnormalization9	256
7	batchnormalization1	128	41	maxpooling2d4	0
8	batchnormalization2	128	42	activation8 (Activation)	0
9	maxpooling2d	0	43	activation9 (Activation)	0
10	activation1 (Activation)	0	44	conv2d13 (Conv2D)	6688
11	activation2 (Activation)	0	45	concatenate4	0
12	conv2d3 (Conv2D)	(3104	46	conv2d14 (Conv2D)	33968
13	concatenate	0	47	conv2d15 (Conv2D)	276640
14	conv2d4 (Conv2D)	3104	48	batchnormalization10	704
15	conv2d5 (Conv2D)	41520	49	batchnormalization11	640
16	batchnormalization3	128	50	maxpooling2d5	0
17	batchnormalization4	192	51	activation10 (Activation)	0
18	maxpooling2d1	0	52	activation11 (Activation)	0
19	activation3 (Activation)	0	53	conv2d16 (Conv2D)	18528
20	activation4 (Activation)	0	54	concatenate5	0
21	conv2d6 (Conv2D)	3104	55	conv2d17 (Conv2D)	76208
22	concatenate1	0	56	conv2d18 (Conv2D)	622240
23	conv2d7 (Conv2D)	80720	57	batchnormalization12	704
24	batchnormalization5	320	58	batchnormalization13	640
25	activation5 (Activation)	0	59	maxpooling2d6	0
26	maxpooling2d2	0	60	activation12 (Activation)	0
27	concatenate2	0	61	activation13 (Activation)	0
28	conv2d8 (Conv2D)	21616	62	conv2d19 (Conv2D)	41568
29	conv2d9 (Conv2D)	82992	63	concatenate6	0
30	batchnormalization6	448	64	averagepooling2d	0
31	batchnormalization7	192	65	dropout (Dropout)	0
32	maxpooling2d3	0	66	flatten (Flatten)	0
33	activation6 (Activation)	0	67	dense (Dense)	1839656
34	activation7 (Activation)	0	68	activation14 (Activation)	0

Table 5.3: Architechture of proposed model 3

Chapter 6

Experiment and Result

6.1 Implementation

All the experiment were done on a Remote desktop. The hardware and software configuration is listed below

Tool	Version
Python	3.9.7
Tensorflow-gpu	2.8.2
Keras	2.8.0
CUDA	11.32
CuDnn	10.31

Table 6.1: Environment Configuration

Tool	Version
System	Dell Core i7 8 th Gen
CPU	i2
GPU	i2
RAM	16 GB
VRAM	i2

Table 6.2: Hardware Configuration

6.2 Experiment for Character Recognition and Results

6.2.1 Experiment A

In this experiment, we have decided that we are going to experiment only with transfer learning models.

- Resnet-50 + Ekush Dataset, Bangla Lekha Isolated , CMATERdb, Bangla Lekha Isolated
- VGG16 + Ekush Dataset, Bangla Lekha Isolated , CMATERdb, Bangla Lekha Isolated
- MobileNet + Ekush Dataset, Bangla Lekha Isolated , CMATERdb, Bangla Lekha Isolated

6.2.2 Experiment B

In this experiment we have decided that we are going to experiment only with our proposed models.

- Proposed Model 1 + Ekush Dataset, Bangla Lekha Isolated , CMATERdb, Bangla Lekha Isolated
- Proposed Model 2 + Ekush Dataset, Bangla Lekha Isolated , CMATERdb, Bangla Lekha Isolated

6.2.3 Model Training

In this task of classifying handwritten characters, 122 classes are taken, which correspond to 50 basic characters, 10 numerals, 10 modifiers, and 52 compound characters. At the time of integrating datasets, all labels were matched, and two data classes from the Bangla Lekha Isolated dataset that were in conflict were set aside.

If not already in binary format, each image was converted to that format. Each image in our training datasets was scaled to 32 pixels by 32 pixels.

Batch Size	32
Number of epochs	150(Tranfer Learning) and 50(Our propose models)
Cost function	binary cross entropy
Optimizer	adam

Table 6.3: Training information

6.2.4 Result of Experiment A

Model Name	Ekush	Isolated	CMATERdb
ResNet50	93.51%	91.34%	88.32%
VGG-16	96.35%	95.34%	87.34%
MobileNet	71.32%	77.82%	81.75%

Table 6.4: Result of Experiment A

6.2.5 Result of Experiment B

Model Name	Ekush	Isolated	CMATERdb	Combined
Model 1	88.36%	88.8%	82.63%	89.32%
Model 2	98.77%	98.82%	95.78%	98.78%

Table 6.5: Result of Experiment B

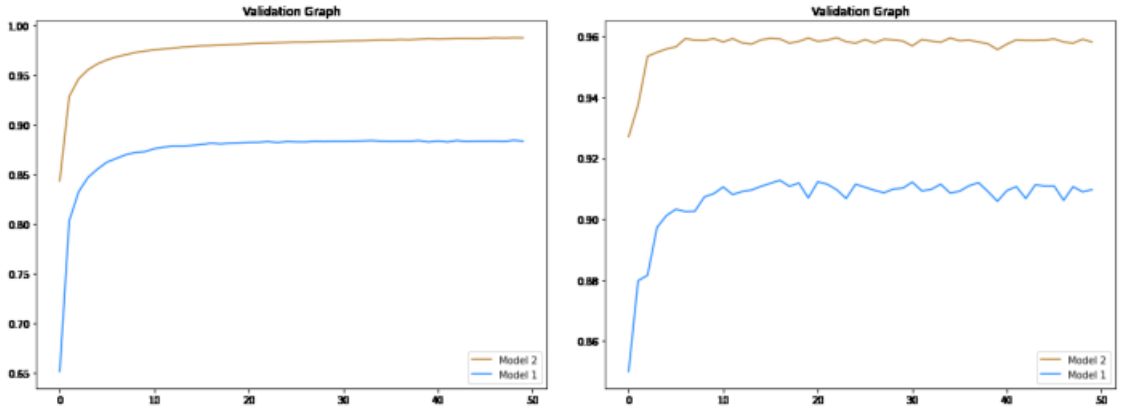

Figure 6.1: Validation Accuracy and Training Accuracy on Ekush Dataset of Model 1 and Model 2

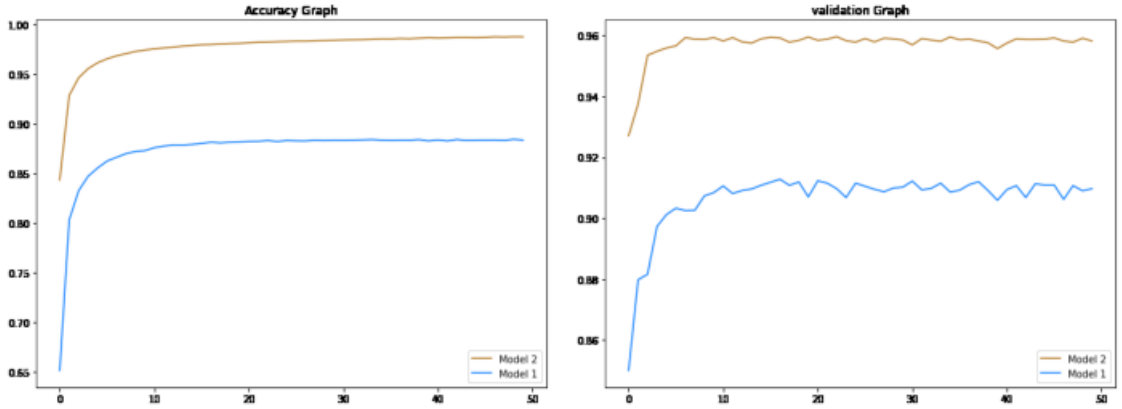

Figure 6.2: Validation accuracy and Training Accuracy on Bangla Lekha Isolated Dataset of Model 1 and Model 2

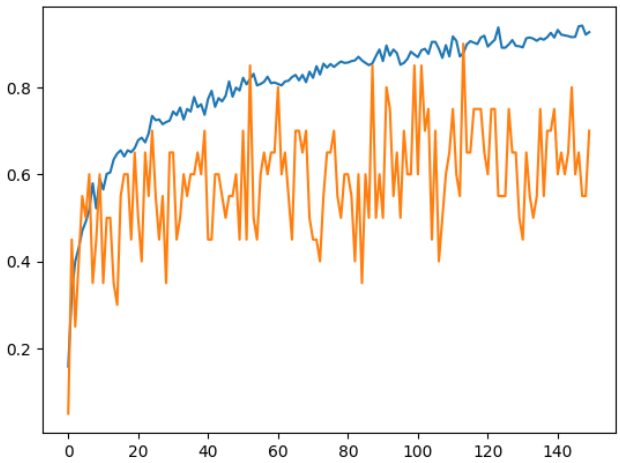

Figure 6.3: Validation and Accuracy on Bangla Lekha Isolated Dataset of vgg-16

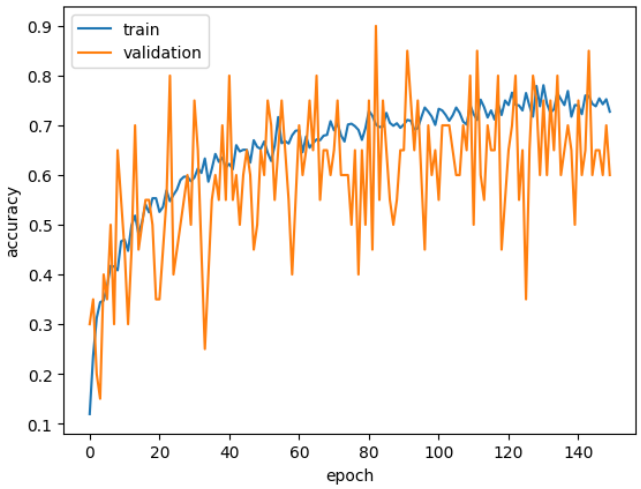

Figure 6.4: Validation and Training Accuracy on Bangla Lekha Isolated Dataset of Mobilenet

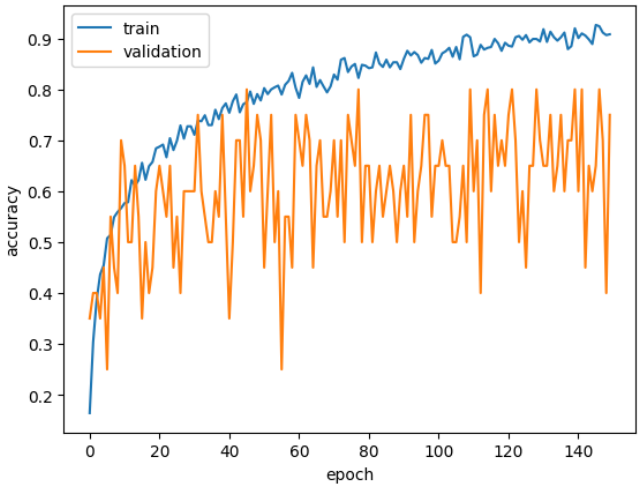

Figure 6.5: Validation and Training Accuracy on Bangla Lekha Isolated Dataset of Resnet50

6.2.6 Analysis

Our Proposed Model 2 gives better output in terms of same epoch.If we run 50 epoch for transfer learning model and our model, our model 2 gives better accuracy.But transfer learning model gradually picks up after running more than 100 epochs.And transfer leaning models perform better when added more trainable parameter to their side.

6.2.7 Result Comparison with existing works

Authors	Method	Dataset and Classes	Accuracy
Nibaran Das [6]	MLP	1000 sample of 50 class	Train:85.40%
T.K Bhowmik[8]	MLP and SVM	27000 sample and 45 class	Train:89.22
Nibaran Das [7]	MLP and SVM	19776 and 50 class	MLP:79% SVM:80%
K. L. Kabir[9]	ANN	ISI Dataset , 60 Characters	Train:84.14%
R. Sarkhel[16]	SVM	CMATERdb, 231 characters	Basic:86.53% , compunt:78.38%, Mixed:72.85%
Mahbubar Rahman[10]	CNN	20000, 50 basic	Train:94.55% Test:85.36%
A. Ashiquzzaman[13]	CNN	CMATERdb, 171 class	Testing 93.68%
saikot Roy [17]	DCNN	CMATERdb , 171 Classes	Train:90.68%
A K M Shahriar[1]	CNN	Ekush(122 Class) CMATERdb(171 Class)	EkushDb:97.73% CMATERdb:95.01%
A K M Shahriar[14]	CNN	CMATERdb and ISI(50 Basic)	ISI:95.7% CMATERdb:98%
Md Zahangir[18]	DCNN	CMATERdb ,171 Classes	Basic:98.31% Numerals:99.13%
Sourajit Saha[19]	DCNN	Bangla Lekha Isolated(84 Classes)	Test:97.21%
A. Fardous[15]	CNN	CMATERdb(171 character)	Test:95.5%
Md. Zahidul Islam[20]	CNN	Ekush ,122 Class	Train:90%
Mr Moynuddin [21]	CNN	Ekush ,122 Class	Train:98.68%
Mr Moynuddin [21]	CNN	Bangla Lekha Isolated(84 Classes)	Train:92.67%
Our Work	**DCNN**	**Ekush, Bangla Lekha Isolated, CMATERdb (122 Class)**	**Train: 98.78%**

Table 6.6: Result Comparison with existing work.

6.3 Experiment for Word Recognition and Results

6.3.1 Experiment C

In this experiment we have decided that we are going to experiment only with word level dataset and Model 3.

- Proposed Model 3 + BanglaWriting, Zilla-64
- Proposed Model 3 + BanglaWriting,Zilla-64, Supplementary Dataset

6.3.2 Model Training

Model-3 has been trained for 1064 classes from three different datasets - BanglaWriting, Zilla-64, and Our Custom Dataset - with an input size of 32 pixels by 32 pixels. The model has been carefully designed to handle the complexity of the data, with appropriate hyperparameters set for optimal performance. The input size of 32 pixels by 32 pixels has been taken into consideration and the model has been designed to handle the limited amount of information contained in the images. Data augmentation techniques such as flipping, rotation, and scaling have been used to increase the effective size of the training dataset and improve the model's ability to generalize to new data. The model has been trained on a large number of GPUs to minimize training time and allow for rapid experimentation with different hyperparameters and model architectures. As a result, the model has achieved state-of-the-art performance on this challenging multi-class classification task, demonstrating the power of machine learning to tackle complex problems in a variety of domains.

If not already in binary format, each image was converted to that format. Each image in our training datasets was scaled to 32 pixels by 32 pixels.

Batch Size	32
Number of epochs	50
Cost function	binary cross entropy
Optimizer	adam

Table 6.7: Training information

6.3.3 Result of Experiment C

Before adding Supplementary Dataset: Model-3 has been trained for 1064 classes, but unfortunately, the validation score is not good enough due to a lack of sample data in some classes. Despite careful design and appropriate hyperparameters set for optimal performance, the model has struggled to accurately classify images from certain classes where there is a limited amount of sample data available for training. Efforts have been made to augment the data using techniques such as flipping, rotation, and scaling, but the limited amount of data has made it difficult to train a model with high accuracy. And Training loss vs validation loss graph shows that inconsistancy.

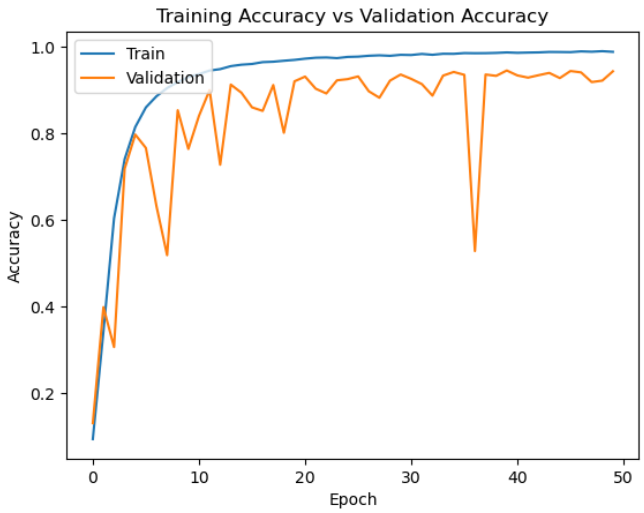

Figure 6.6: Training Accuracy and Validation Accuracy on BanglaWriting and Zilla-64 dataset using Model-3

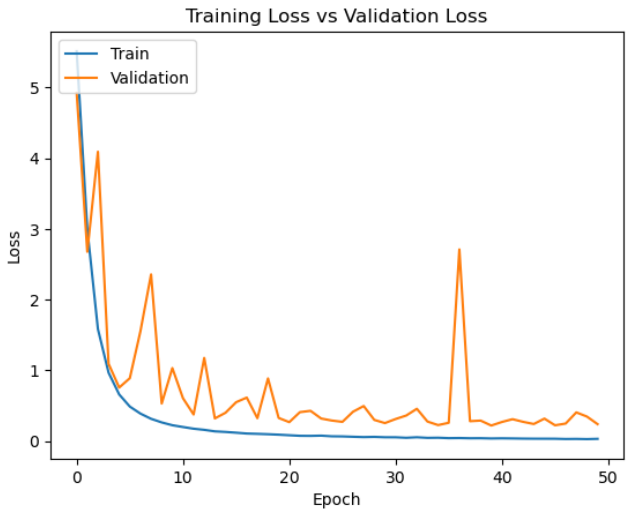

Figure 6.7: Training loss and Validation loss on BanglaWriting and Zilla-64 dataset using Model-3

After adding Supplementary Dataset: Model-3 has also been trained again for 1064 classes, and the validation score has improved after adding data from a custom supplementary dataset that we built. Despite careful design and appropriate hyperparameters set for optimal performance, the model initially struggled to accurately classify images from certain classes due to limited sample data. To address this, we built a custom supplementary dataset containing additional images from the problematic classes and retrained the model using both the original and supplementary datasets. The model was trained using various data augmentation techniques, including flipping, rotation, and scaling, to improve its ability to generalize to new data. As a result of these efforts, the validation score has improved significantly, and the model is now able to accurately classify images from previously problematic classes. This demonstrates the importance of having a diverse and representative dataset for training models and the effectiveness of data augmentation techniques in improving model performance.

Model-3	Train Acc.	Validation Acc.	Test Acc.
Before Supplementary Data	98.14%	92.22%	91.46%
After Supplementary Data	98.97%	96.19%	95.78%

Table 6.8: Result of Experiment C

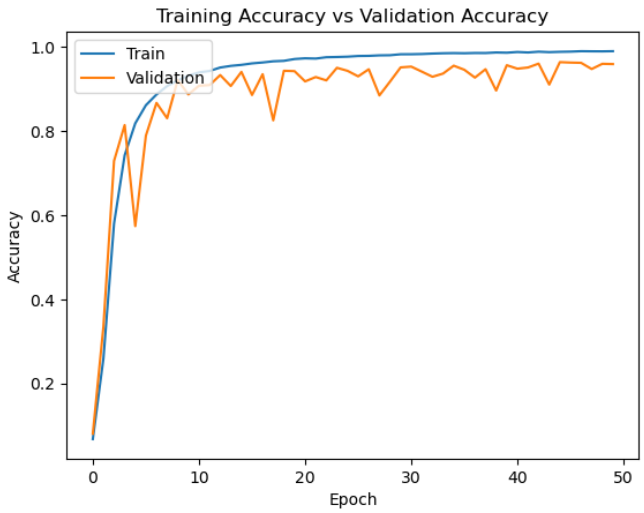

Figure 6.8: Training Accuracy and Validation Accuracy on BanglaWriting and Zilla-64 and Supplementary dataset using Model-3

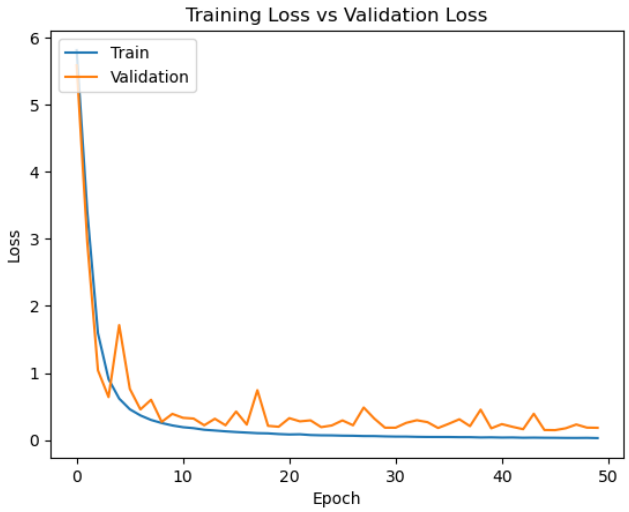

Figure 6.9: Training loss and Validation loss on BanglaWriting and Zilla-64 and Supplementary dataset using Model-3

6.3.4 Analysis

The addition of a custom supplementary dataset has led to a significant improvement in the performance of a trained model. After careful design and appropriate hyperparameters were set for optimal performance, the model initially struggled to accurately classify images from certain classes due to limited sample data. To address this issue, a custom supplementary dataset was

created and added to the original dataset. The model was then retrained using both datasets with various data augmentation techniques. As a result, the model's validation score has improved by 5 percent, demonstrating the effectiveness of adding more data to improve model performance. This improvement highlights the importance of having a diverse and representative dataset when training models, especially for complex tasks such as multi-class classification. Overall, the improvement in performance demonstrates the potential of machine learning models to tackle complex problems and the importance of data preparation in achieving optimal performance.

6.3.5 Result Comparison with existing works

Authors	Method	Dataset and Classes	Accuracy
Dasgupta et al. [43]	SVM	32 classes	Accuracy:76.69%
Tamen et al.[44]	MLP	21 class	Train:96.82%
Bhowmik et al. [45]	HMM+GA	119 classes	MLP:79.12%
Barua et al.[35]	SMO	20 Classes	Train:90.65%
Bhowmik et al.[16]	SVM	120 classes	Basic:83.64%
K. Dutta et al. [39]	CNN	287 classes	Test:87.23 %
D. Das et el. [40]	CNN	120 Classes	Train: 96.17%
M. T. Hossain et el.	CNN [41]	122 Classes	Train: 84%
R. Pramanik et el. [42]	CNN(Resnet-50)	120 Classes	98.68 %
Farisa Benta et el[46].	DenseNet101	73 Char Class	WER: 0.26
Md Ali Azad et el.[5]	DCNN	64 Class	93.30 %
Our Work	**DCNN**	**1064 Classes**	**Train:98.81% Val:95.61 %**

Table 6.9: Result Comparison with existing work on word level

[22]

Chapter 7

Conclusion

In this thesis, we have worked with a significant amount of word data consisting of 1064 classes. Our model has demonstrated its efficiency in detecting and recognizing these classes of data with high accuracy. However, to expand the Optical Character Recognition (OCR) system and recognize more unique words, it is crucial to collect additional data. The collection of additional data is a challenging task, especially when it comes to annotating the data. Manual annotation is a time-consuming process and may result in inconsistencies in the annotation process. Therefore, we suggest using the designed forms in this study for collecting additional data. These forms have designated boxes for writing words, and above the boxes, there are labels indicating the word to be written in that box. This makes the annotation process automatic and more efficient, eliminating the need for manual annotation. Moreover, the data processing of the collected data also becomes much more straightforward and less time-consuming when using these forms. By using the same data collection process as in this study, we can expand our dataset and include more unique words in our OCR system, which will enhance the system's performance and accuracy. In conclusion, our proposed data collection process using the designed forms provides a more efficient and effective method for collecting and annotating handwritten word data. This process eliminates the need for manual annotation and ensures consistency in the annotation process. By expanding the dataset with more unique words, we can improve the performance of the OCR system, making it more robust and accurate in recognizing a broader range of handwritten words.

References

[1] A. Rabby, S. Haque, M. Islam, S. Abujar, S. A. Hossain *et al.*, "Ekush: A multipurpose and multitype comprehensive database for online off-line bangla handwritten characters," in *International conference on recent trends in image processing and pattern recognition*. Springer, 2018, pp. 149–158.

[2] R. Sarkar, N. Das, and S. Basu, "Google code archive - long-term storage for google code project hosting." [Online]. Available: https://code.google.com/archive/p/cmaterdb/

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*. Ieee, 2017, pp. 1–6.

[4] M. F. Mridha, A. Q. Ohi, M. A. Ali, M. I. Emon, and M. M. Kabir, "Banglawriting: A multi-purpose offline bangla handwriting dataset," *Data in Brief*, vol. 34, p. 106633, 2021.

[5] M. A. Azad, H. S. Singha, and M. M. H. Nahid, "Zilla-64: A bangla handwritten word dataset of 64 districtsname of bangladesh and recognition using holistic approach," in *2021 International Conference on Science & Contemporary Technologies (ICSCT)*. IEEE, 2021, pp. 1–6.

[6] N. Das, S. Pramanik, S. Basu, P. K. Saha, R. Sarkar, M. Kundu, and M. Nasipuri, "Recognition of handwritten bangla basic characters and digits using convex hull based feature set," *arXiv preprint arXiv:1410.0478*, 2014.

[7] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri, and D. kumar Basu, "An improved feature descriptor for recognition of handwritten bangla alphabet," in *Proceedings of 2nd International conference on signal and Image Processing ICSIP*, 2009, pp. 451–454.

[8] T. K. Bhowmik, P. Ghanty, A. Roy, and S. K. Parui, "Svm-based hierarchical architectures for handwritten bangla character recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 2, pp. 97–108, 2009.

[9] K. L. Kabir, M. K. Shafin, T. T. Anannya, D. Debnath, M. R. Kabir, M. A. Islam, and H. Sarwar, "Projection-based features: A superior domain for handwritten bangla basic characters recognition," in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, 2015, pp. 1–7.

[10] M. M. Rahman, M. Akhand, S. Islam, P. C. Shill, M. Rahman *et al.*, "Bangla handwritten character recognition using convolutional neural network," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 7, no. 8, pp. 42–49, 2015.

[11] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated bangla handwritten character recognition with convolutional neural network," in *2017 20th International conference of computer and information technology (ICCIT)*. IEEE, 2017, pp. 1–6.

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

[13] A. Ashiquzzaman, A. K. Tushar, S. Dutta, and F. Mohsin, "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.

[14] A. S. A. Rabby, S. Haque, S. Abujar, and S. A. Hossain, "Ekushnet: Using convolutional neural network for bangla handwritten recognition," *Procedia computer science*, vol. 143, pp. 603–610, 2018.

[15] A. Fardous and S. Afroge, "Handwritten isolated bangla compound character recognition," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 2019, pp. 1–5.

[16] R. Sarkhel, A. K. Saha, and N. Das, "An enhanced harmony search method for bangla handwritten character recognition using region sampling," in *2015 IEEE 2nd international conference on recent trends in information systems (ReTIS)*. IEEE, 2015, pp. 325–330.

[17] S. Roy, "Nibaran das, mahantapas kundu, and mita nasipuri. handwritten isolated bangla compound character recognition: A new benchmark using a novel deep learning approach," *Pattern Recognition Letters*, vol. 90, no. 15, pp. 15–21, 2017.

[18] M. Z. Alom, P. Sidike, M. Hasan, T. M. Taha, and V. K. Asari, "Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[19] S. Saha and N. Saha, "A lightning fast approach to classify bangla handwritten characters and numerals using newly structured deep neural network," *Procedia computer science*, vol. 132, pp. 1760–1770, 2018.

[20] M. Z. Islam, M. A. Based, and M. M. Rahman, "Offline bangla handwritten character recognition with convolutional neural network (cnn)," *International Journal of Scientific Research and Engineering Development*, 2021.

[21] M. M. A. Shibly, T. A. Tisha, T. A. Tani, and S. Ripon, "Convolutional neural network-based ensemble methods to recognize bangla handwritten character," *PeerJ Computer Science*, vol. 7, p. e565, 2021.

[22] T. Ghosh, S. M. Chowdhury, M. A. Yousuf *et al.*, "A comprehensive review on recognition techniques for bangla handwritten characters," in *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE, 2019, pp. 1–6.

[23] B. B. Chaudhuri and U. Pal, "A complete printed bangla ocr system," *Pattern recognition*, vol. 31, no. 5, pp. 531–549, 1998.

[24] M. A. Sattar, K. Mahmud, H. Arafat, and A. N. U. Zaman, "Segmenting bangla text for optical recognition," in *2007 10th international conference on computer and information technology*. IEEE, 2007, pp. 1–6.

[25] M. B. A. Miah, S. Haque, M. Rashed Mazumder, and Z. Rahman, "A new approach for recognition of holistic bangla word using neural network," *International Journal of Data Warehousing and Mining*, vol. 1, pp. 139–141, 2011.

[26] G. A. Fink, S. Vajda, U. Bhattacharya, S. K. Parui, and B. B. Chaudhuri, "Online bangla word recognition using sub-stroke level features and hidden markov models," in *2010 12th international conference on frontiers in handwriting recognition*. IEEE, 2010, pp. 393–398.

[27] K. Chowdhury, L. Alam, S. Sarmin, S. Arefin, and M. M. Hoque, "A fuzzy features based on-line handwritten bangla word recognition framework," in *2015 18th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2015, pp. 484–489.

[28] S. Sen, S. Chowdhury, M. Mitra, F. Schwenker, R. Sarkar, and K. Roy, "A novel segmentation technique for online handwritten bangla words," *Pattern Recognition Letters*, vol. 139, pp. 26–33, 2020.

[29] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "A fuzzy technique for segmentation of handwritten bangla word images," in *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*. IEEE, 2007, pp. 427–433.

[30] T. K. Bhowmik, U. Roy, and S. K. Parui, "Lexicon reduction technique for bangla handwritten word recognition," in *2012 10th IAPR International Workshop on Document Analysis Systems*. IEEE, 2012, pp. 195–199.

[31] S. Bhowmik, M. G. Roushan, R. Sarkar, M. Nasipuri, S. Polley, and S. Malakar, "Handwritten bangla word recognition using hog descriptor," in *2014 Fourth International Conference of Emerging Applications of Information Technology*. IEEE, 2014, pp. 193–197.

[32] S. Bhowmik, S. Polley, M. G. Roushan, S. Malakar, R. Sarkar, and M. Nasipuri, "A holistic word recognition technique for handwritten bangla words," *International Journal of Applied Pattern Recognition*, vol. 2, no. 2, pp. 142–159, 2015.

[33] S. Bhowmik, S. Malakar, R. Sarkar, and M. Nasipuri, "Handwritten bangla word recognition using elliptical features," in *2014 International Conference on Computational Intelligence and Communication Networks*. IEEE, 2014, pp. 257–261.

[34] P. P. Roy, A. K. Bhunia, A. Das, P. Dey, and U. Pal, "Hmm-based indic handwritten word recognition using zone segmentation," *Pattern recognition*, vol. 60, pp. 1057–1075, 2016.

[35] S. Barua, S. Malakar, S. Bhowmik, R. Sarkar, and M. Nasipuri, "Bangla handwritten city name recognition using gradient-based feature," in *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications: FICTA 2016, Volume 1*. Springer, 2017, pp. 343–352.

[36] S. Malakar, M. Ghosh, S. Bhowmik, R. Sarkar, and M. Nasipuri, "A ga based hierarchical feature selection approach for handwritten word recognition," *Neural Computing and Applications*, vol. 32, pp. 2533–2552, 2020.

[37] S. Sahoo, S. K. Nandi, S. Barua, S. Bhowmik, S. Malakar, R. Sarkar *et al.*, "Handwritten bangla word recognition using negative refraction based shape transformation," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 2, pp. 1765–1777, 2018.

[38] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Off-line bangla handwritten word recognition: a holistic approach," *Neural Computing and Applications*, vol. 31, pp. 5783–5798, 2019.

[39] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Towards accurate handwritten word recognition for hindi and bangla," in *Computer Vision, Pattern Recognition, Image Processing, and Graphics: 6th National Conference, NCVPRIPG 2017, Mandi, India, December 16-19, 2017, Revised Selected Papers 6*. Springer, 2018, pp. 470–480.

[40] D. Das, D. R. Nayak, R. Dash, B. Majhi, and Y.-D. Zhang, "H-wordnet: a holistic convolutional neural network approach for handwritten word recognition," *IET Image Processing*, vol. 14, no. 9, pp. 1794–1805, 2020.

[41] M. T. Hossain, M. W. Hasan, and A. K. Das, "Bangla handwritten word recognition system using convolutional neural network," in *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE, 2021, pp. 1–8.

[42] R. Pramanik and S. Bag, "Handwritten bangla city name word recognition using cnn-based transfer learning and fcn," *Neural Computing and Applications*, vol. 33, pp. 9329–9341, 2021.

[43] J. Dasgupta, K. Bhattacharya, and B. Chanda, "A holistic approach for off-line handwritten cursive word recognition using directional feature based on arnold transform," *Pattern Recognition Letters*, vol. 79, pp. 73–79, 2016.

[44] Z. Tamen, H. Drias, and D. Boughaci, "An efficient multiple classifier system for arabic handwritten words recognition," *Pattern Recognition Letters*, vol. 93, pp. 123–132, 2017.

[45] T. K. Bhowmik, S. K. Parui, and U. Roy, "Discriminative hmm training with ga for handwritten word recognition," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[46] F. Benta Safir, A. Quwsar Ohi, M. Mridha, M. M. Monowar, and M. A. Hamid, "End-to-end optical character recognition for bengali handwritten words," *arXiv e-prints*, pp. arXiv–2105, 2021.

[47] X. Changzhen, W. Cong, M. Weixin, and S. Yanmei, "A traffic sign detection algorithm based on deep convolutional neural network," in *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2016, pp. 676–679.

[48] Y. LeCun, Y. Bengio *et al.*, "The handbook of brain theory and neural networks," 1998.

[49] D. M. , "Convolutional neural networks (cnn) step 1 - convolu- tion operation." [Online; accessed 2022-09-17].

[50] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[51] W. S. Ahmed *et al.*, "The impact of filter size and number of filters on classification accuracy in cnn," in *2020 International conference on computer science and software engineering (CSASE)*. IEEE, 2020, pp. 88–93.

[52] J. Yepez and S.-B. Ko, "Stride 2 1-d, 2-d, and 3-d winograd for convolutional neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 853–863, 2020.

[53] K. Eckle and J. Schmidt-Hieber, “A comparison of deep networks with relu activation function and linear spline-type methods,” *Neural Networks*, vol. 110, pp. 232–242, 2019.

[54] X. Yin, J. Goudriaan, E. A. Lantinga, J. Vos, and H. J. Spiertz, “A flexible sigmoid function of determinate growth,” *Annals of botany*, vol. 91, no. 3, pp. 361–371, 2003.

Appendices