A Squeeze and Excitation ResNeXt-Based Deep Learning Model for Bangla Handwritten Basic to Compound Character Recognition

by

Mohammad Meraj Khan
16366009

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

Department of Computer Science and Engineering
Brac University
August 2021

Declaration

It is hereby declared that

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

Student's Full Name & Signature:

Mohammad Meraj Khan
Student ID: 16366009

Approval

The thesis titled "A Squeeze and Excitation ResNeXt-Based Deep Learning Model for Bangla Handwritten Basic to Compound Character Recognition" submitted by **Mohammad Meraj Khan (16366009)** of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science & Engineering on 31^{st} of December, 2021.

Examining Committee:

Supervisor:
(Member)

Mohammad Shorif Uddin, PhD
Professor
Department of Computer Science and Engineering
Jahangirnagar University

Co-Supervisor:
(Member)

Zavid Parvez Digitally signed by Zavid Parvez DN: cn=Zavid Parvez, o=Brac University, ou=CSE, email=zavid.parvez@bracu.ac.bd, c=US Date: 2021.12.31 06:50:51 +11'00'

Mohammad Zavid Parvez, PhD
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Examiner:
(External)

Mohammad Nurul Huda, PhD
Professor
Department of Computer Science and Engineering
United International University

Examiner:
(Internal)

Md. Khalilur Rahman, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Examiner:
(Internal)

Muhammad Iqbal Hossain, PhD
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement

Hereby, I Mohammad Meraj Khan consciously assure that for the manuscript "A Squeeze and Excitation ResNeXt-Based Deep Learning Model for Bangla Handwritten Compound Character Recognition" the following is fulfilled:

1) This material is original work, which has not been previously published elsewhere.

2) The manuscript is not being considered for publication anywhere at this time.

3) The writers' research and analysis are reflected in the publication wholly and truthfully.

4) The paper appropriately acknowledges the efforts of co-authors and co-researchers.

5) The findings are discussed in the context of previous and ongoing research.

The norms of the Ethical Statement can have serious implications if they are broken.

I agree to the aforementioned declarations and certify that this submission adheres to Solid State Ionics' rules as described in the Authors' Guide and the Ethical Statement.

List of Publication

- **M.M. Khan**, M. S. Uddin, M. Z. Parvez, L. Nahar, "A squeeze and excitation ResNeXt-based deep learning model for Bangla handwritten compound character recognition," Journal of King Saud University – Computer and Information Sciences, Published on 16 February, 2021, Available online : https://doi.org/10.1016/j.jksuci.2021.01.021

- **M.M. Khan**, M. S. Uddin, M. Z. Parvez, L. Nahar, J. Uddin, "A Deep Convolution Neural Network-Based SE-ResNeXt Model for Bangla Handwritten Basic to Compound Character Recognition," Journal of Hunan university natural sciences, 2021 **(Accepted for Publication)**

Abstract

With the recent advancement in artificial intelligence, the demand for handwritten character recognition increases day by day due to its widespread applications in diverse real-life situations. As Bangla is the world's 7th most spoken language, hence the Bangla handwritten character recognition is demanding. In Bangla, there are basic characters, numerals, and compound characters. Character identicalness, curviness, size and writing pattern variations, lots of angles, and diversity makes the Bangla handwritten character recognition task very challenging. There are few papers published recently which works both Bangla numeral, basic and compound handwritten characters, but the accuracy level in all three areas is not so satisfactory. The main objective of this paper is to propose a novel model which performs equally outstanding in all three different character types and to increase the efficiency to build a real-world Bangla Handwritten character recognition system. In this work, we describe a novel method of recognition for Bangla basic to compound character using a very special deep convolutional neural network model known as Squeeze-and-Excitation ResNext. The architectural novelty of our model is to introduce the Squeeze and Excitation (SE) Block, a very simple mathematical block with simple computation but very effective in finding complex features. We obtained 99.80% accuracy from a bench-mark dataset of Bangla handwritten basic, numerals, and compound characters containing 160,000 samples. Additionally, our model demonstrates outperforming results compared to other state-of-the-art models

Keywords: Bangla handwritten-character recognition, Deep Convolutional Neural Network, Squeeze and Excitation ResNext, Optical character recognition, Global average pooling.

Dedication

To my wife.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional Neural Network

$DCNN$ Deep Convolutional Neural Network

GAP Global Average Pooling

$ILSVRC$ ImageNet Large Scale Visual Recognition Challenge

IPA International Phonetic Alphabet

$LSTM$ Long Short Time Memory

MLP Multi-Layer Perceptron

$MQDF$ Modified quadratic discriminant function

OCR Optical Character Recognition

$ReLU$ Rectified Logical Unit

$ResNet$ Residual Network

$ResNeXt$ Residual Network Next

SE Squeeze and Excitation

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Bangla Language Scripts

Bangla is one of the most spoken languages, with approximately 228 million native speakers and about 37 million second-language speakers. It is the fifth most-spoken native language and the seventh most spoken language by a total number of speakers in the world. Bangla is the official language and mother tung of Bangladesh and the West Bengal State of India. It has a very rich history of more than a thousand years. There are 50 basic characters, 11 numerals, and around 300 compound characters in the Bangla language [38]. The basic character set contains 11 vowels and 39 consonants. Table 1.1 represents the printed version of basic characters and their corresponding International Phonetic Alphabet (IPA) Symbols, Table 1.2 represents the corresponding hand handwritten version. Two or more characters together form a compound character. The printed version of some frequently used compound characters along with their IPA Symbols, and their corresponding handwritten version is shown in Table 1.3 and Table 1.4, respectively.

Character	০	১	২	৩	৪	৫	৬	৭	৮	৯	
IPA Symbol	0	1	2	3	4	5	6	7	8	9	
Character	অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
IPA Symbol	ɔ	a	i	ii	u	uu	rʱ	ɛ	ɛi	o̯	o̯u
Character	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট
IPA Symbol	k	kʰ	g	gʱ	ŋ	ʧ	ʧʰ	dʒ	dʒʱ	n	t
Character	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন	প	ফ
IPA Symbol	tʰ	ɖ	ɖʱ	n	t	tʰ	d	dʱ	n	p	pʰ
Character	ব	ভ	ম	য	র	ল	শ	ষ	স	হ	ড়
IPA Symbol	b	bʱ	m	dʒ	r	l	ʃ	ʃ	ʃ	h	ɖ
Character	ঢ়	য়	ৎ	ং	ঃ	ঁ					
IPA Symbol	ɖʱ	e̯	t	ŋ	ɦ	◌̃					

Table 1.1: Bangla numerals and basic characters and corresponding IPA symbols (1st-row numerals – 3nd-row vowels, 5th-12th rows consonants)

Table 1.2: Handwritten Bangla numerals and basic characters are shown in Table 1.1
(1st-row numerals, 2nd-row vowels, 3rd-6th rows consonants)

Character	ক্ষ	ব্দ	ঙ্গ	স্ক	স্ফ	স্থ	চ্ছ	ক্ত
IPA Symbol	kʰ	bd	ŋg	ʃk	ʃpʰ	ʃh	tʃtʃʰ	ktô
Character	ম্ন	ষ্ণ	স্প	প্ত	ম্ব	ণ্ড	ড্ড	ক্ষ্ম
IPA Symbol	ʃnô	ʃnô	ʃp	ptô	mbô	nɖ	dbʰ	kʰmô
Character	হ্ন	ষ্ঠ	ল্ল	ম্প	ন্দ	ন্ধ	ম্ম	ণ্ঠ
IPA Symbol	nhô	ʃtʰ	lpô	ʃpô	nd	ndʰ	mm	ntʰ

Table 1.3: A printed version of the considered Bangla compound characters and corresponding IPA symbol

Table 1.4: Bangla compound characters' handwritten version that is shown in Table 3(24 classes)

1.2 Compound Character Formation

In most cases, the compound character does not retain the actual shape of the basic characters from which it is made of. Few examples of different compound character

formations are depicted in Table 1.5. In this table, we observe that the position of the basic character which took part to form the compound character also changes the shape. In some cases, the compound character has no similarities with the basic characters. The fourth row in Table 1.5 shows such a compound character "ঔ".If we closely look into the second and third-row where there are two different compound characters,"ক" and "খ" formatted from two basic characters, here the shape of the basic character is partially retained. One important thing to notice here is that even though the same basic character "ও" is common for both compound characters but its shape is different in the resulted compound character. The last row shows a compound character "স্ত্র", its formation is much complex than the rest characters in Table 1.5, where none of the three basic character shapes is retained. In this table, only in the first row, the compound character retains the basic character's shape. Character identicalness, variations, and writing patterns sometimes make different characters similar looking. Table 1.6 shows a few similar-looking different characters.

Combinations		Compound Character
চ + ছ	=	চ্ছ
ষ + প	=	ষ্প
ষ + ণ	=	ষ্ণ
ঙ + গ	=	ঙ্গ
স + ত + র	=	স্ত্র

Table 1.5: Some examples to show the development of compound characters.

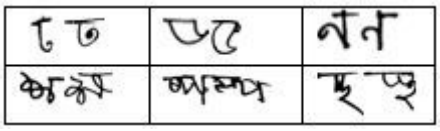

Table 1.6: Some similar-looking Bangla handwritten characters

1.3 Applications

Handwritten character recognition finds widespread applications in diverse fields, such as automation of survey form data entry, vehicle number plate recognition, various documentation digitization, bank cheque processing, OCR applications, etc.

1.4 Challenges and Goals

There are many reasons behind the difficulty of Bangla handwritten characters, such as similarities between characters, writing pattern variations, changed shape in a compound character, too much curviness, variations in sizes and shapes. The length of aces, degree of angles the sizes of turns also make it difficult. The presence of 'Matra' (a horizontal line at the top) even makes it harder. Even a single dot makes a character looks like a different character. Recognition of all (combined numerals, basic and compound) characters is much difficult than only numeric characters, or only basic characters, or only compound characters due to an increase in the number of classes.

There are very few works on combined Bangla handwritten characters compared to only basic or only numerals or only compound [8], [9], [14], [17], [18], [20], [21], [29], [32], [35], [38] handwritten characters recognition.

The most notable techniques used in these works are supported vector machine (SVM) [38], multilayer perceptron (MLP) [20], convolutional neural network (CNN) [8], [9], [14], [18], [21], [29], [32], [35]. The accuracy level and the precision achieved by these models are not up to the mark to build a standard Bangla OCR. In a recent study in the sector of object recognition, a state-of-the-art model shows superior performance, impressed by the result we have decided to explore and for further improvement, we have decided to work on the variety of squeeze and excitation ResNeXt deep convolutional neural network models. This is also a continuation of one of our previous works [41], where we have worked to recognize Bangla handwritten compound characters only. These models are pretty accurate to determine the inter-channel feature dependencies, which in turn makes it efficient to learn very complex features. The main contribution of this work is to develop an effective deep learning technique to recognize a full set of Bangla characters (basic characters, numerals, and compound characters).

1.5 Outline

We have organized the paper in the following manner, Section 2 for similar works. Section 3 for proposed models' architecture. Section 4 for experiment and result analysis. Section 5 for Conclusion.

Chapter 2

Literature Review

2.1 Related Works

As English is the international language, so it is well studied and techniques are already been developed for the English language. But in this paper, our main focus is on the Bangla language. In our research, we have identified there are a lot of works conducted on various languages other than Bangla. Among these, the most mention-able works performed on Hindi, Tamil, Gujrati, Devanagari, Urdu, Arabic, Chinese, Pasto, Japanese, Romans [10], [16], [19], [23], [25], [27], [30], [31], [33], [34], [36], [37], [39], [40]. Some mention-able works are conducted also in Bangla, the most notable ones are discussed in [8], [9], [14], [17], [18], [20], [21], [29], [32], [35], [38]. In most of the early works, researchers depend on the process of hand-engineered feature extraction. But these processes are erroneous, difficult to extract complex features, takes a lot of time and effort, also the outcome is not satisfactory to recognize the characters efficiently.

2.1.1 MLP or SVM based Models

Bhowmik *et al.* [20] investigated a basic MLP (multi-layer perceptron) for the recognition of Bangla handwritten characters. Recently, Bhattacharya *et al.* [21] proposed a two-stage approach using a modified quadratic discriminant function (MQDF) classifier and then an MLP recognizer. Pal *et al.* [17] also worked on the recognition of Bangla compound characters using the MQDF-based recognition technique. Das *et al.* [29] proposed a better approach for the recognition of Bangla basic as well as compound character recognition using an MLP for feature extraction and an SVM (support vector machine) for recognition. Similarly, Basu *et al.* [9] also worked on MLP-based recognition.

MLP Based models are inefficient and redundant. In MLP each neuron from a layer is connected with every other neuron of the next layer, which results in too many weights and leads to overfitting. In an SVM Based model, if the dataset is complex and huge, handling correlation becomes difficult and cannot use spatial information in the recognition and detection tasks.SVM does not perform very well when the data set has more noise, mainly when then target classes are overlapping with other.No only that, if the number of features is more than the training data samples, the SVM will under-perform.

2.1.2 CNN based Models

The models proposed by Ashiquzzaman *et al.* [11] and Fardous *et al.* [28] performs slightly better than PLP or SVM models, both of the models are based on convolutional neural network, there is no requirement of hand engineering to learn new features, rather CNN can learn features automatically itself. So, the introduction of CNN decreases the model's complexity, lessens the chance of human error, and eventually more generalized model by enabling the model's capability to learn complex features. But only by stacking the convolution layer one after another, like in their proposed vanilla convolutional neural network models, does not help a lot, which we can see from the performance of these two models.

2.1.3 ResNet Models

Alif *et al.* [9] and Chatterjee *et al.* [26] both works with the Resnet model which is a special type of deep convolutional neural network model. This model wins the ILSVRC contest in 2015, proposed by Microsoft. The construction of the model is very complex, requires a lot of parameters, and training time is way more than other models. If we compare these two models then we can see that going deep as with the model proposed by Chatterjee *et al.* [26] does not achieve more accuracy. It only increases the parameters and increases the training time and makes the model much heavier. This model is hard to deploy to smart devices.

2.1.4 Divide-Merge and BiLSTM Models

Saha and Saha *et al.* [22] proposed a divide and Merge mapping and Optimal pathfinder, which is a kind of deep CNN-based model. Their model achieves better accuracy than previous other methods, but the recognition of complex compound character its performance is not satisfactory at all. CNN and BiLSTM model proposed by Hasan et. al., requires too many parameters, and computational very complex. Vanishing gradient is a common problem in BiLSTM models, to address the overfitting problem is also a big issue.

2.1.5 Summary

From our literature survey we have seen that the majority of early research works were based on SVM, MLP, Hand engineered feature extraction policy or simple cnn architecture, some later works focused on different cnn architectures, but the performance was not on the high. Most of the works are mainly focused on recognizing only single types of character. In our work, we have used A very special deep convolutional neural network-based hybrid ResNeXt Model with Squeeze and excitation module. It is a challenge because this convolutional neural network model is normally used for image recognition, not for classification.

2.2 Convolutional Neural Network

Fukushima is his research work [2] first proposed the concept and design of the Convolutional Neural Network (CNN). But before that the neural network architecture

was proposed by Hubel [1], Fukushima was inspired by Hubel's work and proposed a hierarchical convolutional neural network model. In later years the model been generalized and used for recognize the digits by Lecun [3] , and to recognize objects from image on various datasets like, CIFAR10, MNIST, ImageNet etc by Ciresan [5]. The following Fig. 2.1 shows a very basic block diagram to recognize digits.

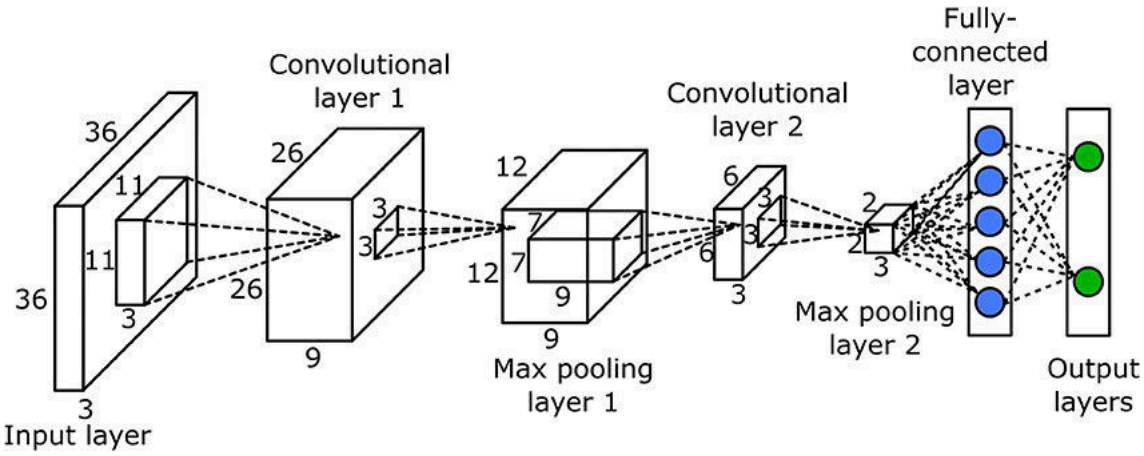

Figure 2.1: A block diagram of a CNN to recognize digits

2.2.1 Convolutional Layer

In a Convolutional Neural Network, the major mathematical operation performed is convolution, which could be seen as the heart of the CNN Model. In earlier research works, the feature extracting was performed mostly by hand engineering, which is a tedious and erroneous task, requires a lot of expertise and domain knowledge. With the introduction of convolution operation, the feature extraction operation has been an automatic process. The operation can be seen as a filter or kernel is stride through the input image, produces matrix multiplication and the sum of those multiplication generates the output image. In the following equation 1, the mathematical notation of the convolutional layer is shown:

$$
\begin{aligned}
&J\text{-}1\ ;\text{-}1 \\
&y = f(XljWlj + \mathrm{b})\ (1) \qquad\qquad (1) \\
&\mathrm{J}{\sim}0;\ {\sim}0
\end{aligned}
$$

Here is input image is denoted by X, Wij denotes the filter, 'i' is the filter height and 'j' is the filter width, So XW is the convolution operation result followed by an addition offset b, then an activation function f is applied to get the output y.

2.2.2 Pooling layers

The main objective of the pooling layer is to pull most of the information without losing any important feature, the output is the sub-sample of the input image. It helps the network not to explode, downside the volume of the model. Generally, the ideal position to use a pooling layer is after a convolutional layer. There are

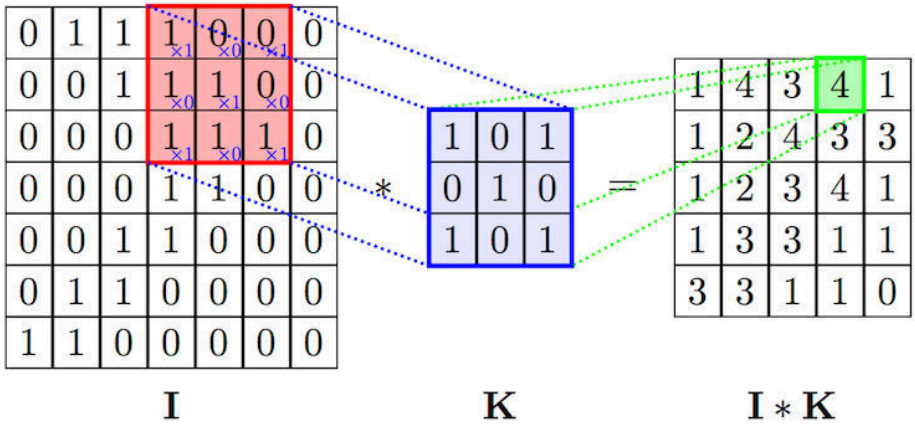

Figure 2.2: Convolution operation

three different pooling layer types, max, min, and average pooling, it depends on the model architecture and the purpose of pooling operation which one it requires. In the following Fig. 2.3 a pooling operation is demonstrated, where a max-pooling later is applied on a 4X4 image, here the pooling window size is 2X2, the output sub-sample 2X2 image is shown on the right-hand side.

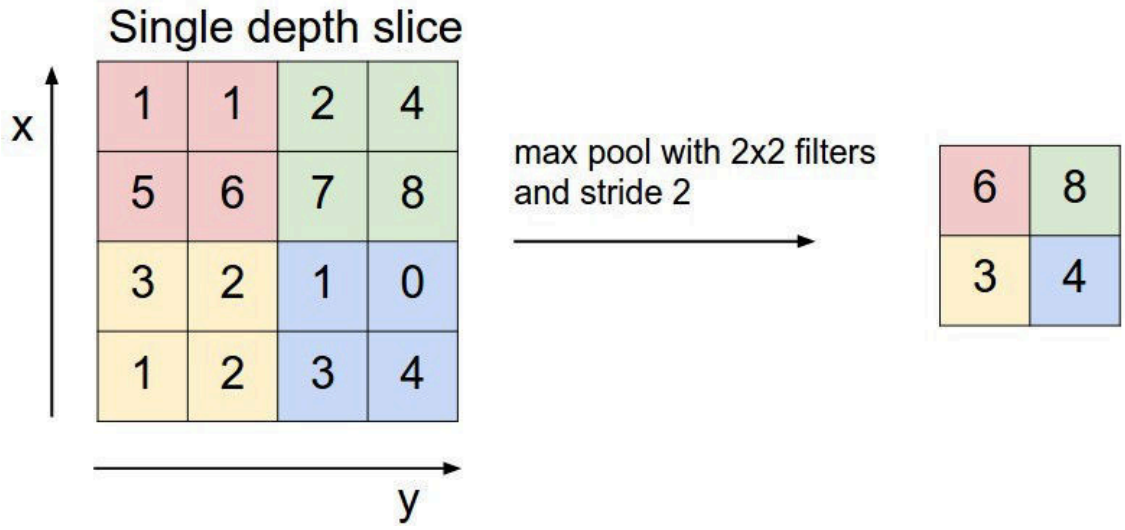

Figure 2.3: Max pooling in CNN

2.2.3 ReLU Nonlinearity

The activation function plays a very important role in the Neural network. It provides the model with a nonlinearity and thresholding the output value of a convolutional or fully connected layer. So, it prevents the model to explode, makes the model more stable. In most real work problems, the boundary line or the co-relations between the data of different categories are nonlinear, the neural network without a nonlinear functionality is just a linear regression model which cannot recognize the different categories well, the performance will become very poor. There is various kind of activation function, but in our research works, we mainly used RELU, Sigmoid, and SoftMax. The position of the activation function depends on

the architecture of the neural network model and the objective of each layer. In our proposed model we used RALU just after a convolutional layer and after a fully connected layer, we used sigmoid after a fully connected layer which is after a RELU. In the final layer of our model, we used Softmax to find the probability of each category.

$$RELU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x >= 0 \end{cases}$$

Figure 2.4: Max pooling equation

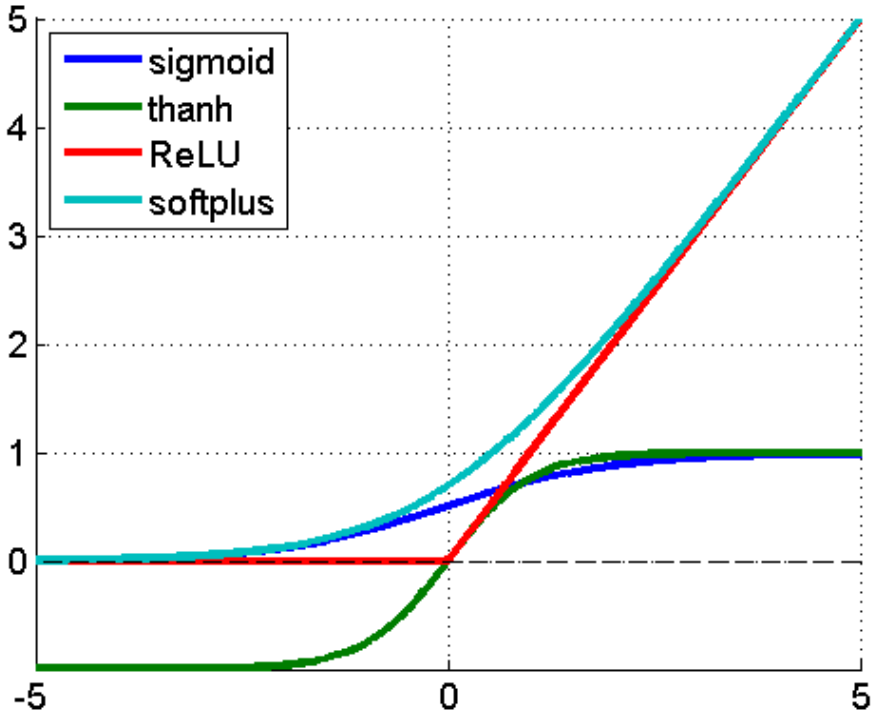

Figure 2.5: Different activation functions

The range of RELU is between zero to positive infinity, any value less than zero will be replaced by zero, from the equation we can see it will take the value which is maximum from zero to input 'X'. For sigmoid, the range is 0 to 1, any value less than 0 will threshold by 0, and any value greater than 1 will be replaced by 1.

2.2.4 Fully connected layers

Every layer in a neural network can be seen as a filter to recognize a specific feature, The initial layers are responsible for high-level features whereas the later or deeper layer is for detecting the complex or low-level features. The purpose of using the fully connected layer is to capture the high-level reasoning. In a fully connected

layer, every neuron from a layer is connected with every other neuron to another layer. There are too many parameters required for a fully connected layer, and the redundancy rate is very high. Overfitting is a common problem in a fully connected layer, so the dropout technique is used to shoot out a few neurons to achieve model generalization.

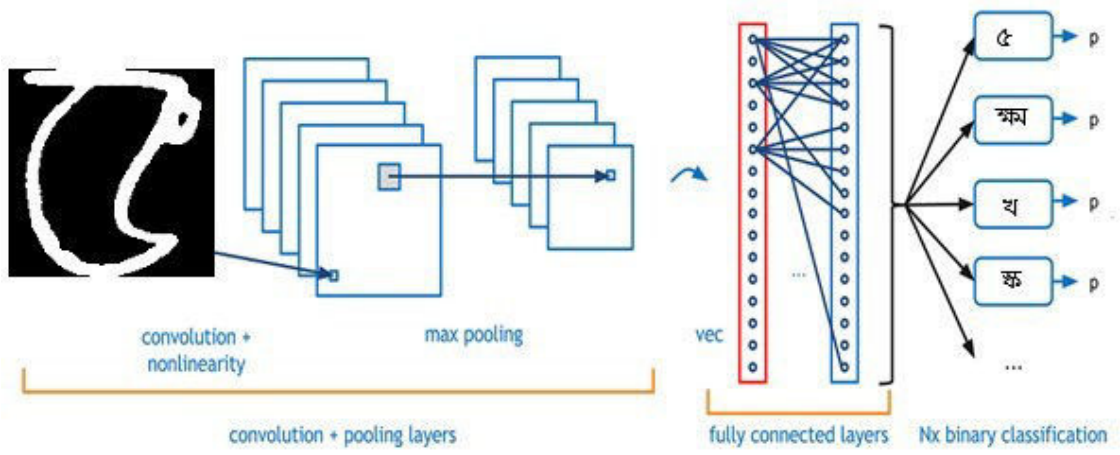

Figure 2.6: Fully connected layers in CNN

In a neural network architecture, one common area where a fully connected layer can be placed is right before the classification layer. In our proposed model we used a fully connected layer in two different places, Firstly, In the SE Block right after the global average pooling layer, we used two fully connected layers, each following an activation function RELU and Sigmoid, and Secondly in the final layer right before the classification layer.

2.3 Optimization

Tuning up the hyperparameter, minimizing the error, and increasing the accuracy is performed by the technique Optimization. By selecting and tuning the Learning rate, the number of batch sizes, and finding the proper combination helps the neural network model perform well.

Some common optimization techniques are Momentum, Adagrad, RMSProm, Gradient Descent, and ADAM optimizer. In our proposed model we used a Gradient descent optimizer, In gradient descent there are three different techniques were used, Stochastic Gradient Descent (SGD) when gradient descent is calculated on a single data, Batch Gradient Descent when applied to total data, and Mini-Batch Gradient descent when applied on a batch data set.

SGD is very easily scalable, computationally efficient, and more stable, There are a few cautions that need to follow when using SGD, the learning rate can not be set to a high value, there is a risk of skipping the proper solution. And if the are multiple local minima, SGD does not work well. But it is a great technique when it requires a faster model optimization.

Equation 2.1 shows the mathematical form of optimization :

$$w = \arg\min_{w} \sum_{\mathbf{x}^{-i} \in X} loss(f(\mathbf{x}^{-}), f_w(\mathbf{x}^{-})) \tag{2.1}$$

The equation is to optimize the loss between actual f(x) and predicted fw(x) output. In each iteration of the input data X, the weight w changes a bit to minimize the loss. Which direction and by what amount of changes in parameter makes the loss decrease more is defined by gradient descent. Equation 2.2 shows the optimization for a minibatch:

$$w_{t+1} = w_t - \eta \; Q(w_t) = w_t - \eta \sum_{i=1}^{N_x} Q_i(w_t) \tag{2.2}$$

where Qi(w) = loss(f(x¯i), fw(x¯i)) is the loss of the model for training sample x¯i ϵ X. The update rule for a minibatch Nx for simplicity, where the accumulated weight for a subset of input data with gradient direction is scaled by the learning rate 'n'. which is then added or subtracted with the current weight to get the next weight. The combined update rule for the entire data sets is defined as below equation 2.3:

$$w_{t+1} = w_t - \eta \; Q_i(w_t) \tag{2.3}$$

Weight in the next step would be the difference between current weight wt and the production of direction parameter, current weight Wt, and η learning rate. It is very costly and inefficient to update the full gradient, we used an approximation of gradient instead of an actual gradient for a minibatch input.

The update rule using Nesterov momentum can be defined as:

$$v_{t+1} = \mu v_t - \eta \; Q_i(w_t + \mu v_t) \tag{2.4}$$

$$w_{t+1} = w_t + v_{t+1}, \tag{2.5}$$

Here momentum is denoted by vt, in subsequent iteration if the direction doesn't change, Nestorov momentum (NM) accumulates the direction. μ denoted coefficient for Nesterov [6], which is ranged from 0 to 1.In each iteration of training, the minibatch training inputs data are shuffled, the update rule is applied once in each epoch of each iteration.

2.4 Loss Function

Loss function plays an important role in the neural network model while in the training phase. It draws a clear line that depicts the difference between the model output and the targeted output. One of the main objectives of the training phase is to minimize the loss and increase the accuracy. To choose a perfect loss function, there are a few criteria that need to fulfill, firstly the function should be continuous, secondly, it should be easily differentiable so that in backward pass the derivatives flow smoothly and finally it should be less computational complexity. There are quite a few loss functions, but mean squire error, cross-entropy loss are in more practical usages than the rest because of the requirement of less computational power, easily differentiability, and continuousness.

For a well-trained model, if we draw a loss function graph, it can be seen that both the training and testing lines are very close to each other and close to almost zero. It expresses two things: first, the model is generalized well, training loss is not too low than the validation loss (no over-fitting), training loss is not too high (no under-fitting), second, if both of the lines close to zero indicates a well-trained stable model.

Cross entropy loss function can be mathematically formulated as below the equation 2.6:

$$\bar{\mathbf{X}}_{\mathrm{N}} \quad y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \tag{2.6}$$

Here the total number of training data is denoted by 'N', from yn = f(x¯n) , where x¯n denotes the input and yn is the targeted result and from yˆn = fw(x¯n) where yˆn denoted actual output.

Chapter 3

Our Proposed Method

3.1 Method Overview

The s architecture of our proposed method is explained in detail in this section. Fig. 3.1 shows the simplified block diagram of our method.

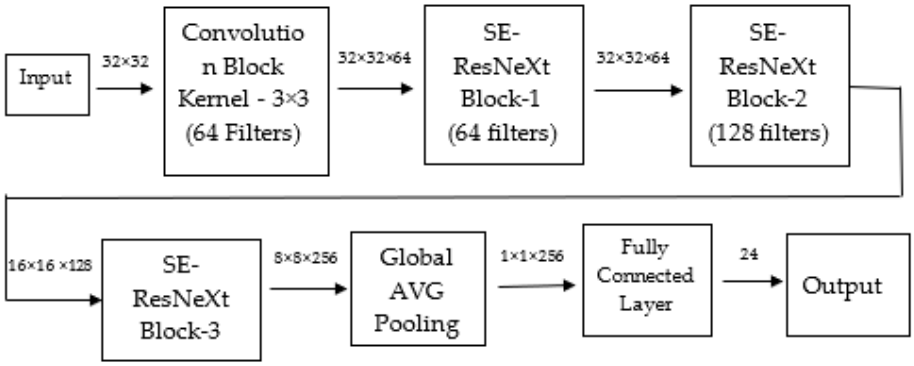

Figure 3.1: Proposed model's block diagram

The proposed model takes a 32×32 size image as input and the final fully connected layer outputs the probabilities of 84 possible classes of 50 basic characters, 10 numerals, and 24 compound characters. The first layer is a convolutional block followed by the first SE-ResNeXt Block, - the second and the third SE-ResNeXt Block, then a global average polling layer followed by the final fully connected layer. The detailed construction of the SE-ResNeXt block is described in detail in the latter of this section. Each of the SE-ResNeXt blocks is a combination of 3 stacked SE-ResNeXt layers.

3.2 Squeeze and Excitation Blocks

The squeeze and excitation (SE) block [26] is a computational unit that can be constructed for any given transformation $F_{tr} : \mathrm{X} \rightarrow \mathrm{U}, \mathrm{X} \in R^{H \times W \times C}$ where F_{tr} denotes a convolutional operator performed on the input X that produces the output U, H and W are the input image's height and width, respectively and C represents the channel. Let $\mathrm{V} = [v_1, v_2, \ldots, v_c]$ denotes the learned set of filter kernels, where

refers to the parameters of the c-th filter.We can then write the outputs of F_{tr} as $U = [u_1, u_2, ..., u_c]$, where

$$u_c = v_c * X = \sum_{s=1}^{c'} v_c^s * x^s \qquad (3)$$

Here vc is the filter kernel, X is the input image, and $\star$ denotes the convolution operation, $v_c = [v_c^1, v_c^2, \dots, v_c^{C'}]$ and $X = [x^1, x^2, \dots, x^{C'}]$, the term bias is not included in the equation for the sake of simplicity. The output is produced by summing up all the operations in all channels.Where v_c^s is a spatial kernel, which operates with the input X for that channel. The channel dependencies are embedded in vc through spatial correlation. The transformation operation which retrieves the informative feature of inter-channel dependencies empowers the network. In Fig. 2 the block diagram of the SE building block is shown, which will be added with the ResNext which is shown in Fig. 3.2.

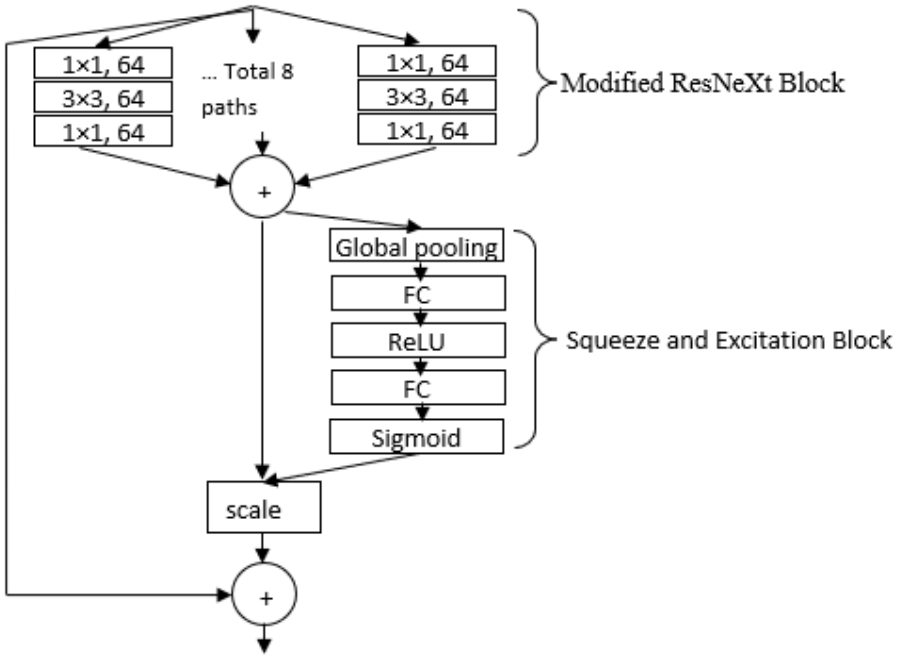

Figure 3.2: SE-ResNeXt Building Block.

3.3 Squeeze: Global Information Embedding

The deep neural network faces an exploitation problem of inter-channel dependencies in the later layers, mainly with the layers where the receptive field size is comparatively smaller. It is not able to share the information or the computational output which is produced by applying a filter to a local receptive field with other connective channels. We calculate a summary for a channel, which can also be viewed as a statistical representation of a channel to address the barrier of information sharing among channels. Global average pooling is the technique that makes this possible,

this operation describes a channel by retrieving the features which carry most of the information. A channel's descriptor $z \in \mathbb{R}^c$, with spatial dimensions H × W can be calculated by the following Equation, through the statistic of a channel z of c-th element can be calculated as:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j) \qquad (4)$$

The convolutional operation between the local receptive field of the input image X and the 2D filter produces the transformation output u_c (image intensity). Here (i, j) denotes the image's coordinate, 'i' could be any value from 0 to H-1, and 'j' could be any value from 0 to W-1.

The whole image information can be retrieved from the transformation output U, which can be seen as the collection of the major information container of the image. This type of technique is widely used in feature engineering [22].

3.4 Excitation: Adaptive Re-calibration

The main objective of the operation in the excitation layer is to fully capture the inter-channel dependencies. To achieve this goal, we can use the channel statistics which are achieved from the previous layer's squeeze operation. To get the desired result, two essential criteria must be fulfilled by a very simple gated layer. these are: (i) the channel-to-channel nonlinearity may be capturable, and (ii) as multiple channels are emphasized as opposed to one hot activation, it must be able to find the channel wise non-mutually-exclusive relationship. In equation (5) this gated function is formulated as:

$$s = F_{ex}(z, W) = \sigma\big(g(z, W)\big) = \sigma\big(W_2 \delta(W_1 z)\big) \qquad (5)$$

Where δ refers to the ReLU [12], [13] function, $W_1 \in R^{\frac{c}{r} \times c}$ and $W_2 \in R^{\frac{c}{r} \times c}$. used in the excitation operation of the above equation. To improve the model's generalization capability and to decrease the complexity, a dimensionality-reduction layer is used. Two fully connected layers parameterized with a gating function around the nonlinearity layer are used. Here a reduction ratio r =16 is applied with the parameter W1, for nonlinearity, a ReLU activation function is used and for dimensionality-increasing, the parameter W2 is used. Finally, a rescaling operation is performed to get the final output.

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c u_c \qquad (6)$$

Where $\tilde{X} = [\tilde{x}_1 \tilde{x}_2, \ldots, \tilde{x}_C]$ and $F_{scale}(u_c, s_c)$ indicates the channel-wise influence of the feature map $u_c \in R^{H \times W}$ and the scalar s_c.

3.5 Dropout

Dropout is a regularization technique; the term denotes dropout randomly selects neurons from both hidden and visible layers in a neural network. This is one of the early techniques to overcome the overfitting problem. In each iteration, as some neurons were dropped out in forwarding pass, then in the backpropagation these neurons do not exist. For the input layer, 0.1 is the recommended dropout rate and for internal layers, this value could be anything between 0.5 to 0.8. Despite some advantages of using dropout, it is needed to be cautious using dropout in some scenarios like: a). regularization is unnecessary when the model is small relative to the volume of the training dataset; B) when we have a limited training time; c) should not use it to the layer before the final classification layer (i.e., last layer), as the model cannot "correct" errors induced by dropout before the classification happens. We have successfully used the dropout technique to address the overfitting problem and to make our model more general. Fig. 3.3 shows a sample network where dropout is applied.

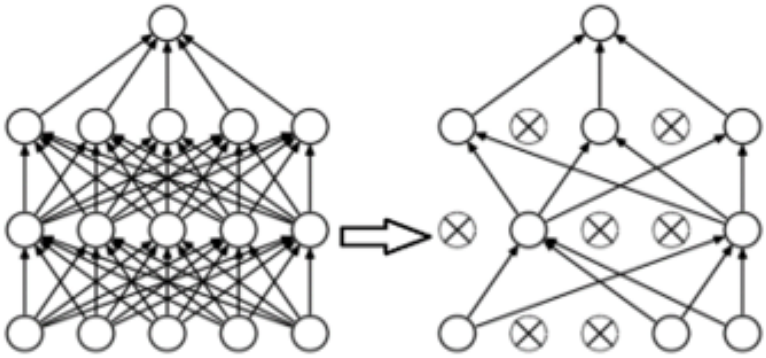

Figure 3.3: Transformation of neural network where dropout [11] in place

3.6 Model Architecture

Stacking layers of SE-ResNext Blocks constructs the model. A slight modification has been performed on the original ResNext model. Here, a simple version of the squeeze and excitation computational layer is added with the existing architecture of ResNext. Three consecutive convolutional layers followed by a ReLU nonlinearity for the first two Conv layers, global average pooling layer, two fully connected layers followed by nonlinearity makes the SE block. In Table 3.1 the model's detailed architecture is shown. The cardinality column shows the splits of the ResNext blocks and the block column shows the number of SE blocks to construct a SE layer. The model takes the input image of 32×32 and outputs a 1×84 single dimensional array of probabilities.

Layer	Kernel Size	Filter	Activation	Cardinality	Block	Output
Conv	3×3	64	RELU	-	-	32×32×64
Conv	1×1	64	RELU	8		32×32×64
Conv	3×3	64	RELU			
Conv	1×1	64	-		3	
Global avg. pooling	-	-	-	-		
FC	-	-	RELU			
FC	-	-	Sigmoid			
Conv	1×1	64	RELU	8		16×16×128
Conv	3×3	64	RELU		3	
Conv	1×1	64	-			
Global avg. pooling	-	-	-	-		
FC	-	-	RELU			
FC	-	-	Sigmoid			
Conv	1×1	64	RELU	8		8×8×256
Conv	3×3	64	RELU			
Conv	1×1	64	-		3	
Global avg. pooling	-	-	-	-		
FC	-	-	RELU			
FC	-	-	Sigmoid			
Global avg. pooling	-	-	-	-	-	256
FC	-	-	-	-	-	84

Table 3.1: The detailed construction and settings of the SE-ResNeXt blocks

Chapter 4

Experimental Setup and Result Analysis

4.1 Dataset Insight and Comparative Study

To train and validate our model, the standard dataset Mendeley BanglaLekha-Isolated 2 dataset [15] has been used. The dataset is well-curated, taken samples from people of different ages, groups, and sex, containing handwritten Bangla numerals, basic and compound characters.

Attributes	MNIST	Mendeley BanglaLekha-Isolated 2
Position	Centered	Non-Centered
Formation	Uniform	Non-Uniform
Classes	10 (only 10 digits)	84 (50 basic characters, 24 compound characters, 10 digits)

Table 4.1: Dataset comparative study - MNIST Vs MENDELEY BanglaLekha-Isolated 2

The dataset contains 10 Bangla numerals, 50 basic characters, 24 different frequently used compound characters. For a single character, around 2000 different handwritten samples were collected and performed pre-processing tasks. The total number of samples is 166106 for 84 different classes in place of 168000, 2000 each, because there are some erroneous samples, which are discarded. For the training and testing, we have taken 165000 samples from this dataset, 132000 for training and 33000 for testing. Mendeley BanglaLekha-Isolated 2 is comparatively complex than the MNIST dataset [36] from scaling, location, and the number of classes' points of view. The differences between these two datasets are shown in Table 4.1.

4.2 Performance Measurement Techniques

The standard equation of Accuracy, Precession, Recall and F1-score are shown below. We can get the accuracy by dividing correctly predicted samples with total samples, get the precision by dividing true positively predicted samples by total predicted positive samples, get the recall by dividing correctly positive predicted

samples by all positive samples. F1-score considers both precision and recall, which is the weighted average of these two values. The performance results are shown in Table 4.2, which are really good performances.

$$Accuracy\ (\%) = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{7}$$

$$Precision\ (\%) = \frac{TP}{TP + FP} \times 100 \tag{8}$$

$$Recall\ (\%) = \frac{TP}{TP + FN} \times 100 \tag{9}$$

$$F1 - score\ (\%) = 2 \times \frac{Precision \times Recall}{Precision + Recal} \times 100 \tag{10}$$

Here, TP denotes the correctly predicted correct value.TN denotes correctly predicted wrong value, FP denotes predicted wrong value as of right. FN denotes predicted right value as wrong.
These measurements can be calculated by Eq. (11) to Eq. (14) [4].

$$TP_i = c_{ii} \tag{11}$$

$$TN_i = \sum_{k=1,k\neq i}^{n} \sum_{j=1,j\neq i}^{n} c_{jk} \tag{12}$$

$$FP_i = \sum_{j=1,j\neq i}^{n} c_{ji} \tag{13}$$

$$FN_i = \sum_{j=1,j\neq i}^{n} c_{ij} \tag{14}$$

Here 'i' denotes the current category and n denotes character categories (n = 84).

4.3 Data Preprocessing

For training and validation of our model, a computer with 9th generation processor Intel 9700K core-i7, 3.6 GHz with 16 GB RAM is used. An NVIDIA GPU RTX-2080Ti with 11GB memory of GDDR6 was used to get the CUDA accelerated parallel computing.

We need to preprocess the images of the dataset before feeding them to our model. All the image sizes are between 155×155 to 185×185. These images needed to be converted to grayscale with a size of 32×32 pixels. We choose 64 as the training batch size. For the testing phase, we choose 10 as an iteration to get the average result.

4.4 Model Performance Observation

We used Nesterov Momentum as the optimization function. We performed batch normalization [24] after a convolutional operation and before feeding the result to the next computational layer and it helps greatly to quick convergence and faster training process. The demonstrated result in both the training and testing phase indicates that the model overcomes the overfitting [7], [12] problem and performs well.

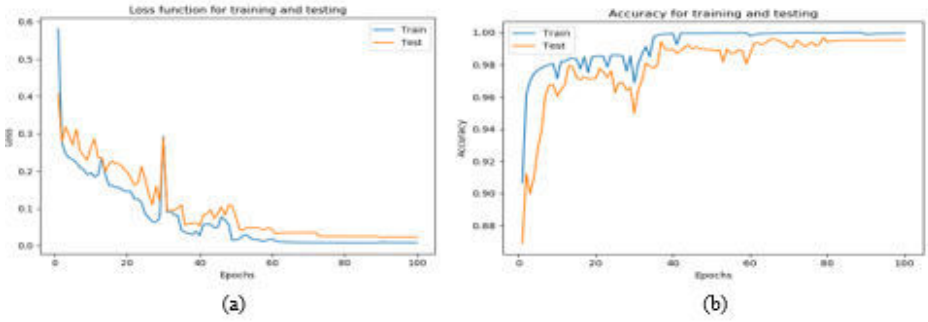

Figure 4.1: A comparative study on model performance in training and testing phase.

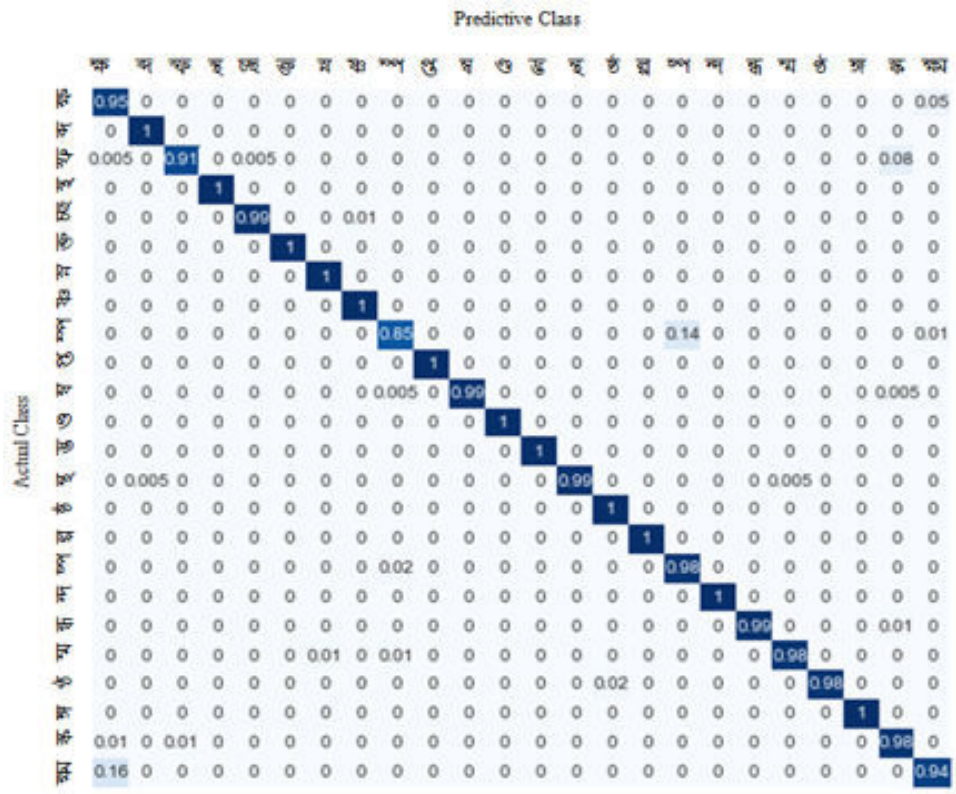

Figure 4.2: Normalized confusion matrix.

To overcome the issue of over-fitting, we used a generalization technique called dropout, which makes the model simpler and more generalized. If we look at Fig. 4.1(a), it is clear that both the training and testing lines are very close to each other

and closes to almost zero. It expresses two things: first, the model is generalized well, training loss is not too low than the validation loss (no over-fitting), training loss is not too high (no under-fitting); second, as both of the lines close to zero indicates a well-trained stable model. From Fig. 4.1(b), we can see that after some iteration both the training and testing accuracy becomes stable, indicates the model is doing well for unseen data. And both of the lines almost close to 1, indicates the model is properly trained. We used another technique called early stopping, which refers to a specific point of training iteration where the training accuracy reaches its peak. After that point, the model would not learn much but loosen its ability to generalize and over-fits the training data. So, for early stopping, the training process should be stopped before passing that point.

Compared to the traditional deep learning model, which requires many hyper-parameters, our proposed model requires to set the cardinality the only one hyper-parameter. We have used RELU and skip connections to address the issue of vanishing gradient. RELU helps efficiently gradient flow where skip connection makes the gradient flow without any loss. Before the final fully connected layer we have used a more native CNN environment layer with the global average pooling, which is very efficient in finding the similarities between the output classes and the input feature maps.

From our experiment, we have seen that the true positive rate is 100 percent for the majority of the classes, except for two classes "৫" and "৬". Where for both classes the true positive rate is 99percent and the false positive rate is 1percent, it is because there are few samples where these two characters look almost similar. If we look closely for these examples, we can see that for character "৫" if the upper portion's arc is open, then it becomes identical to the character "৬".

Some basic characters whose formation and pattern are almost similar to some compound characters and numerals. In those cases, it is harder to distinguish them in obtaining a good result. For example, sample image data of classes "ঠ", "ভ" and "ণ" looks like images of numeric classes "৮", "৫" and compound class "ল্ড" consecutively. In some compound characters, the model faces this problem also, easily noticeable characters of similar patterns are "স্ফ", "ষ্ক", "ষ্প", "ষ্প", "ক্ষ্ম", "ক্ষ". The character "স্ফ" which formation is "স+ফ", and the character "ষ্প", which formation is "ষ+প", but the turns, angels and the writing pattern makes these two characters almost similar to the human eye, our model also find it difficult to distinguish. The same conclusion can be drawn for the character "স্ফ", as it is similar to the character "ষ্ক" and character "ক্ষ্ম" as it is similar to the character "ক্ষ".

We have seen these similarities in some basic characters. If we closely look into the character pairs ("খ", "ঘ"), ("ভ","ড"), ("ণ", "ল্ড") and (" র","ব"), we can find the similarity problem. For the first pair "খ" and "ঘ", even though the computer typed character looks different, but the handwritten format of these two looks similar, samples from some individuals shows that if they don't put the 'matra' (the top horizontal line) then "ঘ" looks like "খ". For characters "ণ" and "ল্ড", if someone puts a 'matra' on top of "ণ" then it looks the same as "ল্ড", which is a compound character. The same can be said for the character "ভ" and "ড". If we look at

character “র” and “ব”, where the dot at the bottom of character “র” is too tiny to identify makes it similar to the character “ব”.

Class	Accuracy	Precision	Recall	F1-score
০	1.00	1.00	1	1.00
১	0.999	1.00	0.990099	1.00
২	1.00	1.00	1	1.00
৩	1.00	1.00	1	1.00
৪	1.00	1.00	1	1.00
৫	0.998	0.99	0.99	0.99
৬	0.998	0.99	0.99	0.99
৭	1.00	1.00	1	1.00
৮	1.00	1.00	1	1.00
৯	0.999	0.99	1	0.99
অ	1.00	1.00	1.00	1.00
আ	1.00	1.00	1.00	1.00
ই	1.00	1.00	1.00	1.00
ঈ	1.00	1.00	1.00	1.00
উ	1.00	0.99	0.87	0.93
ঊ	1.00	0.98	0.99	0.98
ঋ	0.99	0.88	0.71	0.79
এ	1.00	1.00	1.00	1.00
ঐ	1.00	1.00	1.00	1.00
ও	1.00	1.00	1.00	1.00
ঔ	1.00	1.00	1.00	1.00
ক	1.00	1.00	1.00	1.00
খ	1.00	0.95	1.00	0.98
গ	1.00	1.00	1.00	1.00
ঘ	1.00	0.95	1.00	0.98
ঙ	1.00	1.00	1	1.00
চ	0.99	0.89	0.71	0.79
ছ	1.00	1.00	1.00	1.00
জ	1.00	1.00	1.00	1.00

Continued on next page

Table 4.2 – continued from previous page

Class	Accuracy	Precision	Recall	F1-score
ঝ	1.00	0.82	0.77	0.79
ঞ	1.00	1.00	1.00	1.00
ট	1.00	1.00	1.00	1.00
ঠ	1.00	1.00	1.00	1.00
ড	1.00	1.00	1.00	1.00
ঢ	1.00	1.00	1.00	1.00
ণ	1.00	0.98	0.98	0.98
ত	1.00	1.00	1.00	1.00
থ	0.99	0.91	0.70	0.79
দ	1.00	1.00	1.00	1.00
ধ	1.00	0.91	0.90	0.90
ন	0.99	0.95	0.98	0.97
প	1.00	1.00	1.00	1.00
ফ	1.00	1.00	1.00	1.00
ব	1.00	1.00	1.00	1.00
ভ	1.00	1.00	1.00	1.00
ম	0.99	0.91	0.70	0.79
য	0.98	0.95	0.93	0.94
র	0.99	0.98	0.77	0.86
ল	1.00	1.00	1.00	1.00
শ	1.00	1.00	1.00	1.00
ষ	1.00	1.00	1.00	1.00
স	1.00	1.00	1.00	1.00
হ	1.00	1.00	1.00	1.00
ড়	1.00	1.00	1.00	1.00
ঢ়	1.00	1.00	1.00	1.00
য়	0.98	0.92	0.98	0.95
ৎ	1.00	1.00	1.00	1.00
ং	1.00	1.00	1.00	1.00
ঃ	1.00	1.00	1.00	1.00
ঁ	1.00	1.00	1.00	1.00
ক্ষ	0.99	0.84	0.95	0.89

Continued on next page

Table 4.2 – continued from previous page

Class	Accuracy	Precision	Recall	F1-score
ঝ	1.00	1.00	1.00	1.00
ফ্ফ	1.00	0.99	0.91	0.95
ঃ	1.00	1.00	1.00	1.00
চ্ছ	1.00	0.99	0.99	0.99
ক্ত	1.00	1.00	1.00	1.00
ঞ্	1.00	0.99	1.00	1.00
ঞ্চ	1.00	0.99	1.00	1.00
ম্প	0.99	0.96	0.85	0.90
ষ্ট	1.00	1.00	1.00	1.00
স্ব	0.997	1.00	0.99	0.99
ণ্ড	1.00	1.00	1.00	1.00
ড্ড	1.00	1.00	1.00	1.00
হ্	1.00	1.00	0.99	0.99
ষ্ঠ	1.00	0.98	1	0.99
ঞ্	1.00	1.00	1.00	1.00
ম্প	0.99	0.88	0.98	0.92
ন্দ	1.00	1.00	1.00	1.00
ঙ্গ	1.00	1.00	0.99	0.99
ম্য	0.998	0.99	0.98	0.99
ণ্ঠ	1.00	1.00	0.98	0.99
ঙ্গ	1.00	1.00	1.00	1.00
ঙ্গ	1.00	0.91	0.98	0.94
ক্ষ	0.99	0.93	0.84	0.88
Average	99.82%	98.08%	96.90%	97.40%

Table 4.2: Class-wise performance matrix.

From our experimental results, we have drawn an 84×84 normalized confusion matrix. From this confusion matrix and by using the formula from Equations Eq. (7) to (10), we have calculated our proposed model's accuracy, precesion, and other measurements. During the testing phase of our proposed model, we have noticed and recorded the accuracy level for most of the classes reached almost 100, except for some characters, a few of which we have mentioned earlier. To recognize a character the average time taken by our model is 12ms, which denotes that the model can be applied to any real-world scenario.

4.5 Comparative study with Similar State-of-Art Models

Proposed by	Used Methodology	Accuracy	Machine Configuration
Kibria *et al.* [38]	Support Vector Machine (SVM)	88.73%	-
Pramanik *et al.* [20]	Shape decomp. + MLP	88.74%	-
Ashiquzzaman *et al.* [11]	Deep CNN	93.68%	Intel Core-i3, 8GB RAM, Nvidia GTX-1050Ti 4GB DDR-5 GPU
Fardous *et al.* [28]	DCNN with ReLU and Dropout	95.50%	4 CPUs with 61GB of RAM and Nvidia Tesla K80 GPU with 12 GB RAM
Alif *et al.* [9]	ResNet-18	95.99%	Intel Core-i3 (3.30 GHz) CPU with 12 GB RAM and Nvidia 1050Ti 4GB GPU
Chatterjee *et al.* [26]	DCNN + Divide and Merge Mapping + Optimal Path Finder	97.12%	CPU with an NVIDIA 940 GEFORCE 2 GB GPU
Saha and Saha.[22]	ResNet-50	96.12%	-
Alom *et al.* [18]	ResNet + DenseNet	98.31%	Intel Core-i7 CPU 3.33 GHz, 56.00 GB RAM
Hasan *et al.* [29]	DCNN + BiLSTM	98.50%	Intel Core- i7 CPU 3.20GHz, 16GB RAM, and NVIDIA GeForce GTX-1070.
Our Method	SE-ResNeXt	99.82%	Intel Core- i7, 3.6 GHz CPU, 16GB RAM and NVIDIA RTX-2080Ti GPU

Table 4.3: Comparative study – similar recent state-of-the-art methods.

Table 4.3 shows a list of different methods used for Bangla character recognition. It is seen that the performance dramatically improved with CNN or deep CNN-based models. The computational power has indeed increased as days passed, but it mainly accelerates the training process and allows models to train on more data.

From the table, we can see that the performance of the SVM-based [38] model is poor. SVM relies on spectral information only, it cannot use rich spatial information. Moreover, if the data size is large and formation is complex then SVM failed to recognize the underlying correlation. Performance of the MLP (multi-layer perceptron) [20] and shape decomposition-based model is also poor. In MLP every neuron of a layer is connected with every other neuron of another layer, ending up with too many weights, becomes unmanageable, overfitting is a common problem, so the models become less generalized. The shape decomposition model requires hand engineering for feature extraction, which is an erroneous and time-inefficient technique, requires a lot of expertise.

The deep CNN model [11], [28] performs comparatively better than SVM or MLP based model. But the accuracy is not up to the mark for real-life usages. The model proposed by Ashiquzzaman et al. [11] is not properly generalized and fine-tuned. Compare to this, the model proposed by Fardous et al.[28] performs better but is not industry standard.

The performance of both ResNet [9] and DenseNet [18] based models is much better than the models discussed above. Both of these networks require too much memory, even DenseNet requires a bit more than ResNet as it requires to perform concatenation operation on tensors from different layers, but its accuracy is much better. If we look into two ResNet models by Alif et al. [9] and Chatterjee et al. [24] by going deep from 18 to 50 layers does not ensure the significant improvement in accuracy rather makes the model complex and increase the training time.

DCNN + LSTM model [29] performs well. As deep CNN goes deeper by adding layer after layer, means more weights and biases needed to be trained, it requires a lot of training data to get a good result, the training process is extremely expensive. For LSTM overfitting is a very common problem as the dropout technique is a bit difficult to apply in LSTM. LSTM based model has a lot of parameters that require a lot of memories to train. Most importantly LSTM is good for time series or sequential data, but the image is not sequential data.

In our proposed SE-ResnNeXt model, rather than going deep by adding convolutional blocks one after another, we added SE blocks, which makes the model wider with fewer parameters, and is very efficient in learning complex features by fusing channel-wise feature dependencies. It requires only one hyperparameter called split or cardinality, which denotes how many parallel paths we require to operate on the input image or the output of the previous block. As it requires only one hyperparameter to be tuned up, so the model is easily manageable. Another noticeable change is to use skip connections, which allows the gradient to flow naturally without any loss, allows maximum information flow, helps the earlier neuron to train faster. It also addresses the vanishing gradient problem along with the ReLU nonlinearity.

Despite the outperforming result, we have noticed some misclassification in the training phase. We have analyzed and recorded these misclassification samples, most of these samples were very tough to recognize even for the human eye due to the similarities, writing patterns, or erroneous data. The less amount of data,

lack of pattern variations, and formations could have led to this misclassification. Modification or further improvements of models, train on more versatile data could bring further good results. In Fig. 13, few misclassification samples are listed.

Test Data	True Class	Predicted Class
	ঠ	৮
	ভ	৫
	ক্ষ	ক্ষ
	ন্দ	ন্দ

Table 4.4: Some misclassification cases

From Table 4.4, the test character from the first row is predicted as " ৮" which is a numeral character, but its true class is "ঠ", which is a basic character. If we look at some samples from our dataset, "  ", "  " and "  " of class "ঠ", then it becomes clear that these samples are almost identical with the writing samples "", "" and "  " of class" ৮". The same can be seen for the rest of the test character data of the table.

We have compared the misclassification rate and the level of our model with related other mentionable models. Some of these misclassified results by other methods are shown in Table 4.5.

If we look at the misclassification result from Table 4.5, The model proposed by Rabby *et al.* [21] shows poor performance, from the sample image data we can see that images are clear to identify their true class easily. For the first two rows of sample data, the model fails to recognize a very simple but essential feature the 'matra' (the top horizontal line), If we closely look into these two sample characters then we can see the first one has half matra and the second one has no 'matra', but the model's predicted class for these two-sample data shows full matra. The same scenario happened for the model proposed by Alif *et al.* [9]. The models from Pramanik *et al.* [20] and Ashiquzzaman *et al.* [11] perform better and these two models failed to recognize complex images due to formation and cursive-ness. The performance of the model from Hasan *et al.* [29] is even better than all the above, here the sample images are not only complex in terms of formation and curviness but also erroneous. But the last one and the middle sample images are easily identifiable, which the method failed to identify.

Method	Test Data	True Class	Predicted Class
Rabby *et al.* [21]		খ	ঘ
		ঙ	ভ
		য়	র
Alif *et al.* [9]		ঙ	ল্ভ
		ভ	ড
		ড	ভ
Pramanik *et al.* [20]		গ্ন	গ্ন
		ঙ্ক	ঙ্ক
		হ্ম	হ্ম
Ashiquzzaman *et al.* [11]		স্থ	চ্ছ
		প্ন	প্র
		প্র	ত্র
Hasan *et al.* [29]		প্র	স্র
		ণ্ট	ন্ট
		ল্ভ	ন্ত

Table 4.5: Some misclassification cases for other related methods.

Chapter 5

Conclusion

5.1 Our Findings

We have developed a single model with an average accuracy of 99.82% to recognize Bangla basic to frequently used compound characters that can perform equally well in all different types of characters. In this model, rather than going deep in layers and complex architectures, we simply stacked a simple mathematical block, SE-Block, which is very efficient in terms of complex feature identification and computational cost. Quick convergence, fewer chances of overfitting, simplicity. Bangla Handwritten character dataset scarcity is a big barrier to study further on this domain.

5.2 Implication and explanation of findings

By identifying the complex features from inter-channel feature dependencies help our model perform better. The mathematical operation behind the SE block is very simple, enabling the model to consume less computational power, and ensures higher accuracy. Besides the models do not go deep, so it does not require training a lot of parameters, fewer chances of overfitting, and require minimal training time.

5.3 Strengths and limitations

Quick convergence, high accuracy in all three different types of characters, and the requirement of fewer training data sets are major strengths of our proposed model. Architectural simplicity and less computational cost also make our model suitable for industry standards. There are more symbols and special characters in Bangla Language scripts, those are not investigated by our model. Our model is trained only a single dataset. Therefore, more extensive experimentation is required. We also have considered adding IPA (International Phonetic Alphabet) symbol for each character, which will help researchers from foreign languages to study and work on this domain.

5.4 Conclusion

In our paper, we proposed a deep convolutional neural network-based model, named SE-ResNeXt for the recognition of the full set of handwritten Bangla basic, numerals, and compound characters. SE layer is very simple to design and with almost no computational cost. It empowers the model to quick convergence while training and learning channel-wise feature inter-dependencies that are effective in learning complex features. A standard benchmark dataset is utilized to validate the performance of the proposed model. The experimental result demonstrates an average accuracy of 99.82%, a precision of 97.75%, a recall of 97.63%, and an F1-score of 97.62% using the Mendeley BanglaLekha-Isolated 2 dataset. In addition, the proposed model shows comparatively better results than the existing model with higher accuracy.

5.5 Future Study

In the future, we will try to extend our model to separate and recognize each character and symbol from a handwritten document. For a complete Bangla language, there are more than 300 compound characters. However, we worked on only 24 frequently used Bangla compound characters, 50 basic characters, 10 digits. For this reason, a huge amount of work has been left that requires to create a database of around 400 characters. So if we can work around that dataset, considering all prefixes and suffixes then it will be more effective. We could use the help of a deep machine learning technique, especially a generative adversarial network to generate a vast Bangla handwritten character dataset.

Appendix A: The Implementation for Bangla Handwritten Compound Character Recognition Module

A squeeze and excitation ResNeXt-based deep learning model for Bangla handwritten compound character recognition module.

Documentation for the Implementation

In-detailed documentation for the repository and how each module is designed are discussed here.

Implementation Language, Framework and Training Environment

We choose Python to implement our proposed model, it has been a proven programming language for especially in the machine learning and data science domain. Python is very easy to learn and implement, it does not require too much expertise to the research-oriented task. We have used the TensorFlow framework to build our SE-ResNext Deep neural network model, here in place of TensorFlow there are a few other frameworks like Keras, Torch, Theano that could also be used. But TensorFlow provided more control over the building mechanism of the model and the training process. There are a few other packages we have used to preprocess tasks on datasets, like TfLearn, Skimages, NumPy, SciPy, Pickle, etc. For Charting and plotting, we have used the library Matplotlib, PIL, seaborn, IPython.Display library. To train our model we used a machine with a GPU setup, RTX 2080Ti. Convolution operation (matrix Multiplications with stride) is the core operation for any Convolutional Neural Network, which requires parallel processing to speed up the training process. In the training process of our model, we have achieved parallel processing by using the GPU, which is powered with 4352 CUDA cores.

Repository Details

The hierarchy of the files and folders and the location of the images and models are described in this section. There are three main python scripting files: imageprocessing.py, buildmodelTrain.py, processloaddata.py. And the folders are TrainImages,

TestImages, Result, model, logs. We used 80-20 rules to split the entire images from our dataset, 80% of the images go in the folder TrainImages, and the rest 20% of images go in the Testimages folder. For simplicity, we have just uploaded a few sample images in both TrainImages and Testimages folders. If someone wants to train the model on the actual dataset, it is requested to download the MENDELEY BanglaLekha-Isolated 2 Dataset for the authorized site.

The following figure shows the repository hierarchy:

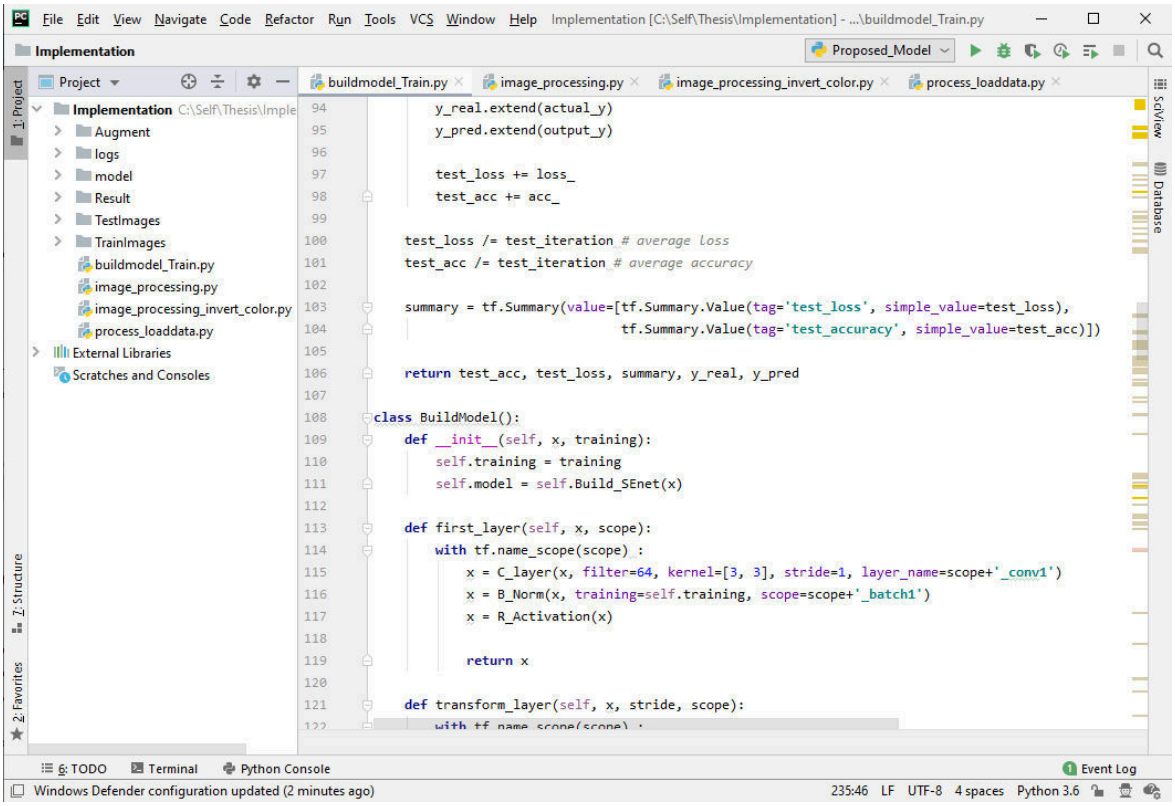

Figure A1: Project outline

We store the iteration-wise result in the folder Result, the trained models are stored in the folder models after a few periods of iterations. Various accuracy of model training observation parameters is stored in the logs folder. We choose batch size 64, the total iteration is 782 and the total epochs are 100. For the testing phase, the iteration was set to 10. The batch size can be higher or lower depending on the configuration of the computer, A GPU with high memory can set a bigger batch size which will take less training time, but a GPU with less memory can set a smaller batch size, will take longer training time.

All the image preprocessing tasks are done in the imageprocessing.py file. All the images from our dataset are not with same height and width. Also, the number of images for the different classes is not the same, so we need some preprocessing tasks before feeding the training and testing data to our model. We had to perform image augmentation to make the same number of images for each class, otherwise, the model will suffer selection biases. The main operations performed here is resizing the image to 64x64, that is the input size to our model. 15-degree rotation, gray scaling

(so the image turns into a single channel instead of 3 RGB channel, which requires less mathematical operation), inverting colors (black background with white text color, wherein the original dataset it was black text color with white background), it is computationally less intensive. Finally, we perform the linearization of the image data, here we convert 64 by 64 images to 4096 by 1 vector. Then the image is ready to feed our model.

The following figure shows some code snippets from the imageprocessing.py file:

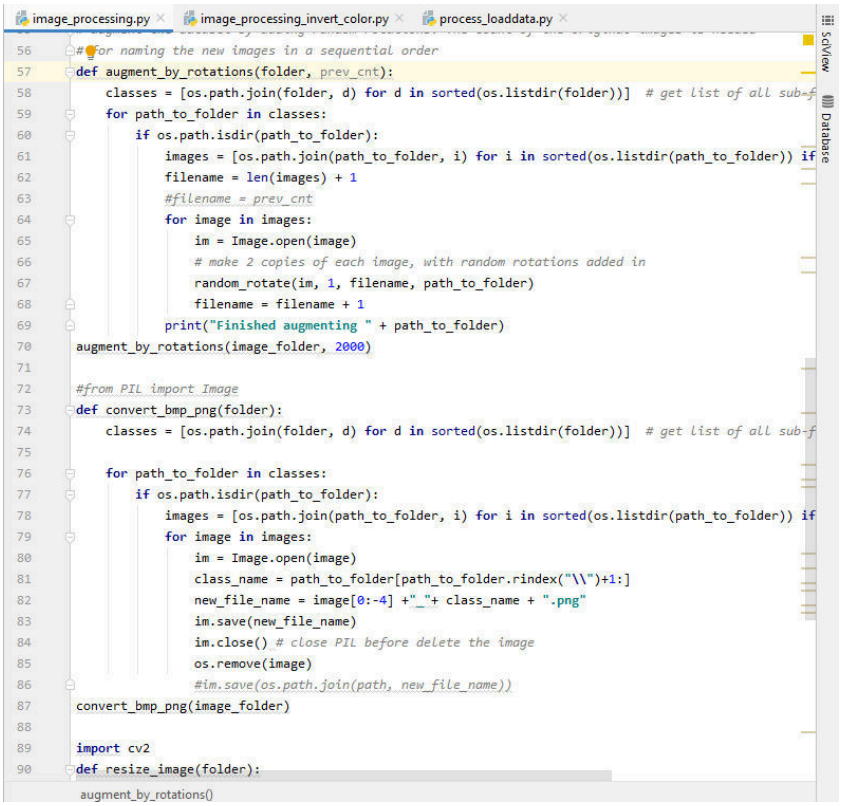

Figure A2: Image processing code snippets

The main functionality to build the model resides in the "buildmodelTrain.py" script. Class SE-ResNext is the class to build the network architecture, here we have methods called "firstlayer" responsible to take the input image, then performing the convolutional operation, followed by batch normalization and RELU activation function for nonlinearity. Some other functions in this class are "transformlayer", "transitionlayer", "split-layer", "selayer", and "residuallayer". The major operations performed in these functions are convolutional operation, global average pooling, average pooling, batch normalization, RELU, Sigmoid, Concatenation, fully connected operation, we have respective methods for all these operations in this script.

The following figure shows some code snippets from the "buildmodelTrain.py" file:

While we feed the training of testing data, it requires shuffling the respective data and label. For labeling the image we used one hot encoding technique. Where it is a vector of size 84x1, as the total number of classes is 84 for our model. The table

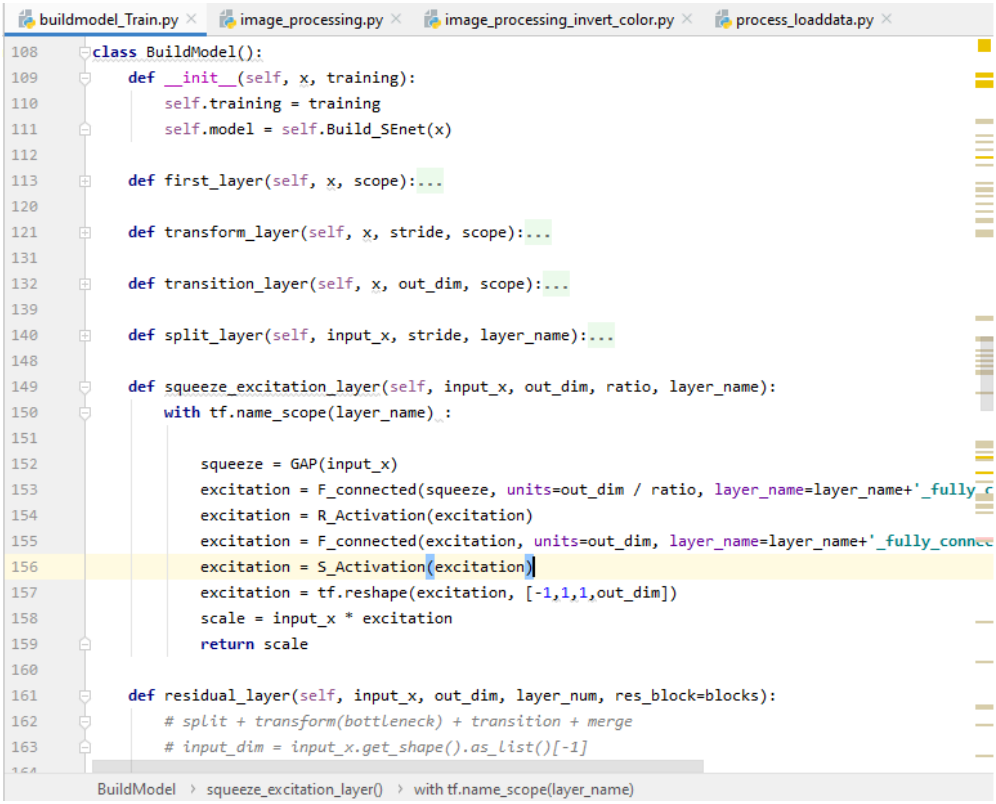

Figure A3: Model Build and training code snippets

below shows the character and their corresponding label or class.

Character	০	১	২	৩	৪	৫	৬	৭	৮	৯	
Label	0	1	2	3	4	5	6	7	8	9	
Character	অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
Label	10	11	12	13	14	15	16	17	18	19	20
Character	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট
Label	21	22	23	24	25	26	27	28	29	30	31
Character	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন	প	ফ
Label	32	33	34	35	36	37	38	39	40	41	42
Character	ব	ভ	ম	য	র	ল	শ	ষ	স	হ	ড়
Label	43	44	45	46	47	48	49	50	51	52	53
Character	ঢ়	য়	ৎ	ং	ঃ	ঁ					
Label	54	55	56	57	58	59					
Character	ক্ষ	ব্দ	ঙ্গ	ঙ্ক	স্ফ	স্থ	চ্ছ	ক্ত	ন্ন	ষ্ণ	ম্প
Label	60	61	62	63	64	65	66	67	68	69	70
Character	প্ত	ন্ম	ণ্ড	ড্ড	ক্ষ্ম	ন্থ	ষ্ঠ	ল্ল	ম্প	ন্দ	ঙ্ক
Label	71	72	73	74	75	76	77	78	79	80	81
Character	ষ্ম	ণ্ঠ									
Label	82	83									

Figure A4: Characters and their corresponding label

The training and evaluation functionality also resides inside the "buildmodelTrain.py" script. We used a Tensorflow saver to save the trained model after certain iterations. So that if some interruptions occur during the training phase, we don't need to train the model from the beginning, we can simply load the last model saved with all the trained parameters. Evaluate function is responsible to evaluate (accuracy and loss in each iteration and epochs) the training and testing phase and monitoring the activity.

Appendix B: Performance Monitoring and Visualization

In our work, we have used epoch-wise loss and accuracy comparison for training and testing phases. It gives us a clear idea of how our model is performing in each epoch in both training and testing. It shows whether the model is overfitting, underfitting, and the status on model convergence, model generalization on unseen data, etc. We performed both combined for 84 classes and separate (numeral-10, basic-50, and compound characters - 24) comparisons for loss and accuracy of training and testing phase. In each case, the model shows an outstanding output.

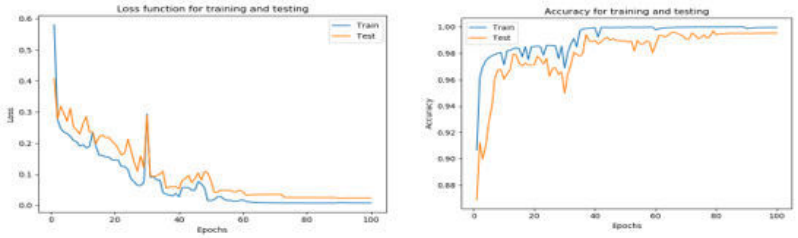

Figure B1: The Lost and the accuracy function in training and testing phase for numerals

For numerals the model shows a quick convergence, the loss is very minimal almost tends to zero and accuracy is almost to one (on a scale of zero to one for both loss and accuracy). The tiny gap between the lines in both training and testing is an indicator that the model is well generalized, not overfitting not underfitting.

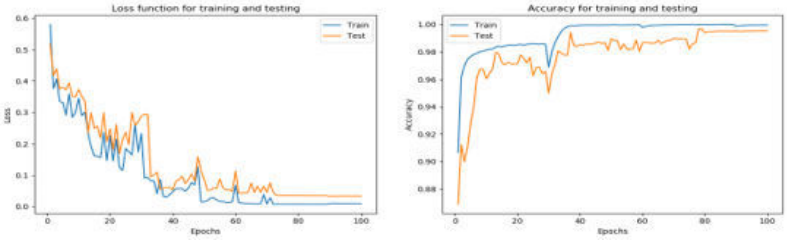

Figure B2: The Lost and the accuracy function in training and testing phase basic character type

For Basic characters, the convergence is also very good but it takes a few more epochs than numerals to reach the same convergence level. This is expected as there are 50 different classes compared to numerals and the formation is more complex as well. Again, the gap between the lines is also minimal for both accuracy and loss graphs

in training the testing phase.

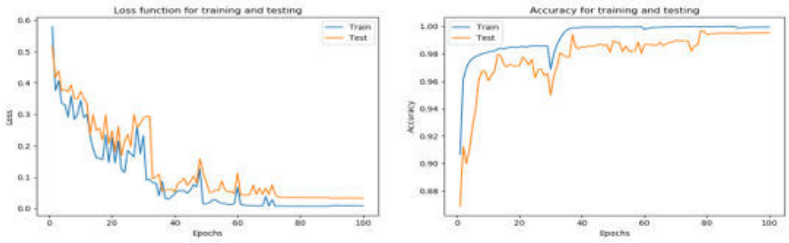

Figure B3: The Lost and the accuracy function in training and testing phase for the compound character type

For compound characters, the convergence comes a bit later than the basic characters. The gap between the line is a bit far away than the other two types, it's due to the variations, similarity, and complex formations of compound character, which makes the model train in the later iterations.

The combined comparisons for all character types (total 84 class: numeral-10, basic-50, and compound characters - 24) are discussed in the main body in chapter 4 Experimental Setup and Result Analysis in section 4.4 Model performance Observation. Where it shows the model is performing equally well in combined types.

Bibliography

[1] D. H. Hubel and T. N. Wiesel, "Effects of monocular deprivation in kittens," *Naunyn-Schmiedeberg's Archives of Pharmacology, Springer*, vol. 248, pp. 492–497, 1964. DOI: https://doi.org/10.1007/BF00348878. [Online]. Available: https://link.springer.com/article/10.1007/BF00348878.

[2] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics, Springer*, vol. 36, no. 4, pp. 193–202, 1980. DOI: https://doi.org/10.1007/BF00344251. [Online]. Available: https://link.springer.com/article/10.1007/BF00344251.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, 1998, pp. 2278–2324. DOI: https://doi.org/10.1109/5.726791. [Online]. Available: https://ieeexplore.ieee.org/document/726791.

[4] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," *ATT Labs*, vol. 2, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[5] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649. DOI: https://doi.org/10.1109/CVPR.2012.6248110. [Online]. Available: https://arxiv.org/abs/1202.2745.

[6] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, PMLR*, vol. 28, 2013, pp. 1139–1147. DOI: https://dl.acm.org/doi/10.5555/3042817.3043064. [Online]. Available: https://proceedings.mlr.press/v28/sutskever13.html.

[7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR*, 2016. [Online]. Available: https://arxiv.org/abs/1511.07289.

[8] M. Shopon, N. Mohammed, and M. A. Abedin, "Bangla handwritten digit recognition using autoencoder and deep convolutional neural network," in *International Workshop on Computational Intelligence (IWCI 2016), (IEEE Explore)*, 2016, pp. 64–68. DOI: https://doi.org/10.1109/IWCI.2016.7860340. [Online]. Available: https://ieeexplore.ieee.org/document/7860340.

[9] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated bangla handwritten character recognition with convolutional neural network," in *International Conference of Computer and Information Technology (ICCIT)*, 2017, pp. 1–7. DOI: https://doi.org/10.1109/ICCITECHN.2017.8281823. [Online]. Available: https://ieeexplore.ieee.org/document/8281823.

[10] A. Ashiquzzaman and A. K. Tushar, "Handwritten arabic numeral recognition using deep learning neural networks in imaging," in *IEEE International Conference on Imaging, Vision Pattern Recognition (icIVPR) (icIVPR)*, 2017, pp. 1–4. DOI: https://doi.org/10.1109/ICIVPR.2017.7890866. [Online]. Available: https://ieeexplore.ieee.org/document/7890866.

[11] 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 *International Conference on Research in Computational Intelligence and Communications Networks (ICRCICN)*, 2017, pp. 1–4. DOI: https://doi.org/10.1109/ICRCICN.2017.8234497. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8234497.

[12] F. Chollet, "Xception: Deep learning with depth wise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: https://doi.org/10.1109/CVPR.2017.195. [Online]. Available: https://arxiv.org/abs/1610.02357.

[13] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. DOI: http://doi.org/10.1109/CVPR.2018.00745. [Online]. Available: https://arxiv.org/abs/1709.01507.

[14] P. Keserwani, T. Ali, and P. P. Roy, "A two phase trained convolutional neural network for handwritten bangla compound character recognition," in *Ninth International Conference on Advances in Pattern Recognition (ICAPR)*, 2017, pp. 1–7. DOI: https://doi.org/10.1109/ICAPR.2017.8592983. [Online]. Available: https://ieeexplore.ieee.org/document/8592983.

[15] N. Mohammed, S. Momen, A. Abedin, *et al.*, "Banglalekha-isolated," *Journal of Data in Brief*, vol. 12, pp. 103–107, 2017. DOI: http://doi.org/10.17632/hf6sf8zrkc.2. [Online]. Available: https://data.mendeley.com/datasets/hf6sf8zrkc/2.

[16] M. A. Mudhsh and R. Almodfer, "Arabic handwritten alphanumeric character recognition using very deep neural network," *Information, MDPI*, vol. 8, no. 3, p. 105, 2017. DOI: http://doi.org/10.3390/info8030105. [Online]. Available: https://www.mdpi.com/2078-2489/8/3/105.

[17] P. R. Sarkar, D. Mishra, and G. R. Manyam, "Improving isolated bangla compound character recognition through feature-map alignment," in *Ninth International Conference on Advances in Pattern Recognition (ICAPR)*, 2017, pp. 1–4. DOI: https://doi.org/10.1109/ICAPR.2017.8593008. [Online]. Available: https://ieeexplore.ieee.org/document/8593008.

[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, no. 4, pp. 1–13, 2018. DOI: https://doi.org/10.1155/2018/6747098. [Online]. Available: https://www.hindawi.com/journals/cin/2018/6747098.

[19] M. Jangid and S. Srivastava, "Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods," *Journal of Imaging, MDPI*, vol. 4, no. 2, pp. 1–4, 2018. DOI: http://doi.org/10.3390/jimaging4020041. [Online]. Available: https://www.mdpi.com/2313-433X/4/2/41.

[20] R. Pramanik and S. Bag, "Shape decomposition-based handwritten compound character recognition for bangla ocr," *Journal of Visual Communication and Image Representation*, vol. 50, no. 4, pp. 123–134, 2018. DOI: https://doi.org/10.1016/j.jvcir.2017.11.016. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1047320317302250.

[21] A. S. A. Rabby, S. Haque, M. S. Islam, S. Abujar, and S. A. Hossain, "Bornonet: Bangla handwritten characters recognition using convolutional neural network," in *International Conference on Advances in Computing and Communication (ICACC-2018)*, 2018, pp. 1–4. DOI: https://doi.org/10.1016/j.procs.2018.10.426. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918321240.

[22] S. Saha and N. Saha, "A lightning fast approach to classify bangla handwritten characters and numerals using newly structured deep neural network," in *International Conference on Computational Intelligence and Data Science (ICCIDS 2018)*, vol. 132, 2018, pp. 1760–1770. DOI: https://doi.org/10.1016/j.procs.2018.05.151. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918308858.

[23] M. M. Saufi, M. A. Zamanhuri, N. Mohammad, and Z. Ibrahim, "Deep learning for roman handwritten character recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12(2), no. 4, pp. 455–460, 2018. DOI: http://doi.org/10.11591/ijeecs.v12.i2.pp455-460. [Online]. Available: http://ijeecs.iaescore.com/index.php/IJEECS/article/view/14425/0.

[24] V. Thakkar, S. Tewary, and C. Chakraborty, "Batch normalization in convolutional neural networks-a comparative study with cifar-10 data," in *International Conference on Emerging Applications of Information Technology (EAIT)*, 2018. DOI: https://doi.org/10.1109/EAIT.2018.8470438. [Online]. Available: https://ieeexplore.ieee.org/document/8470438.

[25] P. V. Bhagyasree, A. James, and C. Saravanan, "A proposed framework for recognition of handwritten cursive english characters using dag-cnn," in *International Conference on Innovations in Information and Communication Technology (ICIICT)*, 2019, pp. 1–4. DOI: https://doi.org/10.1109/ICIICT1.2019.8741412. [Online]. Available: https://ieeexplore.ieee.org/document/8741412.

[26] S. Chatterjee, R. K. Dutta, D. Ganguly, K. Chatterjee, and S. Roy, "Bengali handwritten character classification using transfer learning on deep convolutional neural network," in *International Conference on Research in Computational Intelligence and Communications Networks (ICRCICN)*, 2019, pp. 1–4. [Online]. Available: https://arxiv.org/abs/1902.11133.

[27] T. Clanuwat, A. Lamb, and A. Kitamoto, "Kuronet: Pre-modern japanese kuzushiji character recognition with deep learning," in *International Conference on Document Analysis and Recognition (ICDAR2019)*, 2019, pp. 1–4. DOI: https://doi.org/10.1109/ICDAR.2019.00103. [Online]. Available: https://arxiv.org/abs/1910.09433.

[28] A. Fardous and S. Afroge, "Handwritten isolated bangla compound character recognition," in *International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–4. DOI: https://doi.org/10.1109/ECACE.2019.8679258. [Online]. Available: https://ieeexplore.ieee.org/document/8679258.

[29] M. J. Hasan, M. F. Wahid, and M. S. Alom, "Bangla compound character recognition by combining deep convolutional neural network with bidirectional long short-term memory," in *International Conference on Electrical Information and Communication Technology (EICT)*, 2019, pp. 1–7. DOI: https://doi.org/10.1109/EICT48899.2019.9068817. [Online]. Available: https://ieeexplore.ieee.org/document/9068817.

[30] M. Husnain, M. M. S. Missen, S. Mumtaz, *et al.*, "Recognition of urdu handwritten characters using convolutional neural network," *Applied Sciences, MDPI*, vol. 9, no. 13, p. 2758, 2019. DOI: http://doi.org/10.3390/app9132758. [Online]. Available: https://www.mdpi.com/2076-3417/9/13/2758.

[31] S. Puria and S. P. Singh, "An efficient devanagari character classification in printed and handwritten documents using svm," *Procedia Computer Science*, vol. 152, pp. 111–121, 2019. DOI: http://doi.org/10.1016/j.procs.2019.05.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S18770509193068541.

[32] S. Reza, O. B. Amin, and M. Hashem, "Basic to compound: A novel transfer learning approach for bengali handwritten character recognition," in *International Conference on Bangla Speech and Language Processing (ICBSLP)*, 2019, pp. 1–7. DOI: https://doi.org/10.1109/ICBSLP47725.2019.201522. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9084040.

[33] M. AarifK.O and S. Poruran, "Ocr-nets: Variants of pre-trained cnn for urdu handwritten character recognition via transfer learning," *Procedia Computer Science*, vol. 171, pp. 2294–2301, 2020. DOI: http://doi.org/10.1016/j.procs.2020.04.248. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920312400.

[34] M. S. Amin, S. M. Yasir, and H. Ahn, "Recognition of pashto handwritten characters based on deep learning," *Sensors, MDPI*, vol. 20, no. 20, p. 5884, 2020. DOI: http://doi.org/10.3390/s20205884. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/33080880/.

[35] R. Basri, M. R. Haque, M. Akter, and M. S. Uddin, "Bangla handwritten digit recognition using deep convolutional neural network," in *International Conference Computing Advancements (ICCA 2020), ACM*, 2020, pp. 1–7. DOI: https://doi.org/10.1145/3377049.3377077. [Online]. Available: https://dl.acm.org/doi/10.1145/3377049.3377077.

[36] M. Eltay, A. Zidouri, and I. Ahmad, "Exploring deep learning approaches to recognize handwritten arabic texts," *IEEE Access*, vol. 8, no. 4, pp. 89 882–89 898, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.2994248. [Online]. Available: https://ieeexplore.ieee.org/document/9091836.

[37] J. Gan, W. Wang, and K. Lu, "Compressing the cnn architecture for in-air handwritten chinese character recognition," *Pattern Recognition Letters*, vol. 129, no. 4, pp. 190–197, 2020. DOI: https://doi.org/10.1016/j.patrec.2019.11.028. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167865519303502.

[38] M. R. Kibria, A. Ahmed, Z. Firdawsi, and M. A. Yousuf, "Bangla compound character recognition using support vector machine (svm) on advanced feature sets," in *IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 1–4. DOI: 10.1109/TENSYMP50017.2020.92306091.

[39] Z. Li, Q. Wu, Y. Xiao, M. Jin, and H. Lu, "Deep matching network for handwritten chinese character recognition," *Pattern Recognition*, vol. 107, no. 4, pp. 1–13, 2020. DOI: https://doi.org/10.1016/j.patcog.2020.107471. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0031320320302740.

[40] J. Pareek, D. Singhania, and R. R. K. S. Purohit, "Gujarati handwritten character recognition from text images," *Procedia Computer Science*, vol. 171, pp. 514–523, 2020. DOI: http://doi.org/10.1016/j.procs.2020.04.055. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092031022X.

[41] M. M. Khan, M. S. Uddin, M. Z. Parvez, and L. Nahar, "A squeeze and excitation resnext-based deep learning model for bangla handwritten compound character recognition," *Journal of King Saud University – Computer and Information Sciences*, vol. 2021, no. 4, pp. 1–13, 2021. DOI: https://doi.org/10.1016/j.jksuci.2021.01.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821000392.